NEALT

Northern European Association for Language Technology

NEALT Proceedings Series Vol. 24

Proceedings of the Workshop on

Constraint Grammar - Methods, Tools and Applications

NODALIDA 2015

May 11-13, 2015
Institute of the Lithuanian Language
Vilnius, Lithuania

Proceedings of the Workshop on “Constraint Grammar - methods, tools and applications” at NODALIDA 2015

May 11-13, 2015
Institute of the Lithuanian Language
Vilnius, Lithuania

Editors

Eckhard Bick & Kristin Hagen

Published by
Linköping University Electronic Press, Sweden
Linköping Electronic Conference Proceedings No. 113
URL: http://www.ep.liu.se/ecp_home/index.en.aspx?issue=113
ISSN: 1650-3686
eISSN: 1650-3740
ISBN: 978-91-7519-037-2
NEALT Proceedings Series 24

Cover design Nils Blomqvist

Contents

Preface

The NoDaLiDa CG workshops, occurring since 2005, have become a tradition and an integral part of research exchange for the CG community, reflecting the Nordic roots of Constraint Grammar and the fact that several of the most active CG research groups are located in Nordic countries. Though the field is mature enough that CG papers could also fit into the context of the main conference, we feel that in addition to this, there is still a need for a somewhat less formal forum such as a workshop, with a focused group of participants.

Apart from the traditional field of corpus-oriented tagging and parsing, some of the most dynamic fields in CG research right now are arguably machine translation and tools for less-resourced languages. Thus, in addition to ongoing work at GramTrans, the open source MT initiative Apertium has begun to use Constraint Grammar, and CG is flourishing for many minor languages, such as the Sami languages, Greenlandic, Basque, Tibetan and the Celtic languages. Both of these fields share a need for high-quality lexical, morphological and semantic resources as input to CG grammars and applications. We therefore also invited contributions concerning research in fields relevant to the CG framework on the input side, such as finite-state analyzers, ontologies etc. Finally, we hoped for methodological contributions, exploiting advances in expressive power in the most widely used CG compiler, CG3.

All in all, we thought that there was a sound basis for a workshop in the area, and our hopes were confirmed by high quality of the submitted papers, whose number almost became a problem for the planned time frame of a half-day workshop. Encouragingly, the workshop contributions covered as many languages as there were papers (7), with forages into very diverse language families. Although the expected focus on machine translation did not manifest itself this time, the range of topics was considerable and included both low and high levels of linguistic analysis (morphology, syntax, anaphora), and - on the theoretical side - research on both mathematical, methodological and machine learning issues (SAT, FST and grammar optimization).

We would like to thank the NoDaLiDa organizers for an inspiring and efficient cooperation, and our program committee for their thorough and expert work in the triple reviewing of all submitted papers.

On behalf of the workshop organizers
Eckhard Bick & Kristin Hagen

Workshop organizers

- Eckhard Bick, Research lector, Institute of Language and Communication, University of Southern Denmark
- Tino Didriksen, Developer, GrammarSoft ApS
- Kristin Hagen, Senior engineer, Tekstlaboratoriet, University of Oslo
- Kaili Müürisep, Senior research fellow, Institute of Computer Science, University of Tartu
- Trond Trosterud, Assistant professor in Sámi computational linguistics, University of Tromsø

Program Committee

- Lene Antonsen
- Eckhard Bick (Chair)
- Kristin Hagen
- Kaili Müürisep
- Anders Nøklestad
- Trond Trosterud

Workshop website

http://www.hf.uio.no/iln/om/organisasjon/tekstlab/aktuelt/arrangementer/2015/cg-nodalida-2015.html

Different Issues in the Design and Implementation of a Rule Based Grammar for the Surface Syntactic Disambiguation of Basque

Jose Mari Arriola Egurrola
UPV/EHU University of the Basque Country
josemaria.arriola@ehu.eus

Abstract

This paper presents the results of a set of preliminary experiments reusing some of the modules for the surface syntactic processing of Basque in order to improve the surface syntactic disambiguation. The general idea is to reuse the existing modules at different stages of processing and to find the better order of application of those modules. It aims at introducing a strategy for surface syntactic disambiguation in Basque via rule-based grammars. The results from an evaluation of this disambiguation strategy on a sample corpus are described.

1 Introduction

We will describe some practical issues raised during the design of the strategy for a rule based grammar implemented by means of VISL CG3 (Didriksen, 2010). This work is undertaken in the frame of the Constraint Grammar formalism (Karlsson et al., 1995), and focuses on the design of a disambiguation module which involves both morphology and syntax. The results of a set of preliminary experiments for the design and evaluation of the rule based grammar are presented in order to improve the surface (Abney, 1997) syntactic disambiguation module.

The general framework for the syntactic processing of Basque is composed by several modules (see Figure 1). The IXA research group[1] is working on a robust parsing scheme that provides syntactic analysis in an incremental fashion. Information contained in the EDBL lexical database for Basque (Aldezabal et al., 1999; 2001) constitutes the basis for our analyzers. Once textual input has been tokenized, morphologically analyzed by Morfeus, (Alegria et al., 1996; 1997) and disambiguated by means of Eustagger (Aduriz et al., 2003), syntactic information is added in three distinct stages of processing: i) a CG grammar assigns the syntactic functions to each word-form and deals with the morphosyntactic ambiguity; ii) a CG disambiguation grammar is applied to disambiguate the syntactic functions; iii) a chunk parser provides a partial constituent analysis; and finally, iv) a dependency parser establishes the dependency links (see Figure 1).

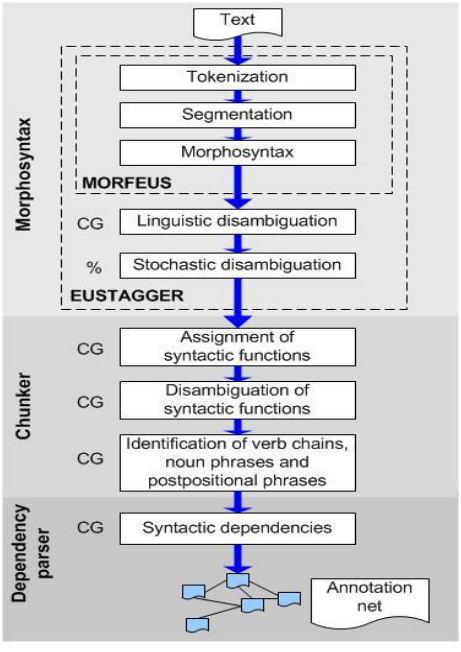

Fig. 1: General framework

We will focus on the first task in the overall parsing scheme, namely the improvement of the syntactic disambiguation process when assigning the syntactic functions. When the syntactic disambiguation grammar module

[1] URL: ixa.si.ehu.es

was first developed, the CG-2 parser (Tapanainen, 1996) was used to implement the rules. There was a main grammar containing morphosyntactic disambiguation rules and purely syntactic disambiguation rules. In a second stage, the main grammar was split into two subgrammars one for morphosyntactic disambiguation and the other for syntactic disambiguation.

Being a language with rich morphology, the basis for the syntactic processing is morphology. The surface oriented syntactic function tags (Karlsson et al., 1995) are assigned by the same module that adds the morphological information to the lemma. The idea is that morphology and syntax are closely related and in a number of cases the syntactic function can be unambiguously tagged to the morphological representation, for instance the ergative plural suffix *–ek* is always subject (@SUBJ). Most of the syntactic information is first introduced with all ambiguities regardless the context and later select and remove rules take care of disambiguation. Both morphological and syntactic ambiguities exist, i.e. one word can receive multiple analyses. Morphological ambiguity includes e. g. part of speech ambiguity e.g. typically noun/verb or noun/adjective. For agglutinative languages there are additional sources of ambiguity (number, case, etc.)

Our aim is to improve the analysis by making use of the information of some of the existing modules, and taking into account that the order of application of the different modules is very important. In this paper we explore a new combination strategy of the modules and respective influence of reordering those modules.

2 Related work

In the literature, we find several approaches to improve syntactic disambiguation. There are three prominent tendencies in disambiguating-grammars and in syntactic analyzers: those based on linguistic descriptions, those based on statistical techniques, and finally, hybrid methods, which combine both. Morphologically rich languages present new challenges, as the use of state of the art parsers for more configurational and non-inflected languages like English does not reach similar performance levels in languages like Basque, Greek or Turkish (Nivre et al., 2007).

We consider the correct morphological disambiguation as a basis for the surface syntactic processing (DoleŽalova, J. and PetkeviČ, V. 2007). In the same direction (Agirre et al, 2012) revealed that the most relevant information is the case carried by the noun and the transitivity of the verb. Besides, (Bengoetxea et al., 2012) shows that POS errors harm the parser.

Ambiguity arises from previously done morphological analyses, and hence, it is closely dependent on decisions made at the morphological level. If only categorial (POS) ambiguity is taken into account, there is an average of 1.55 interpretations per word form, which rises to 2.65 when the full morphosyntactic information is taken into account, giving an overall 64% of ambiguous word-forms. We chose the CG formalism for our purposes of starting to handle syntax. In fact, there are several works that show that good results have been obtained when parsing with CG (Karlsson et al., 1995; Samuelsson and Voutilainen, 1997; Tapanainen and Järvinen, 1997; Bick, 2000).

We have moved from CG-2 to CG3 taking into account the bigger expressive power of CG3 and the open source philosophy. VISL CG3 has been extended to many languages[2].

3 Some experiments

The original CG grammar rules for disambiguation were written in CG-2 and now have been reimplemented and expanded with CG3. In the experiments we explore how much we can improve our syntactic analysis by means of exploiting the interrelation between the different modules. Firstly, some attempts in that direction will be described more specifically in the grammar for morphosyntactic disambiguation. We have included a module for lexical correction for the treatment of complex postpositions and we have tried to benefit from the analysis of the

[2] http://beta.visl.sdu.dk/constraint_grammar_languages.html

chunker. Afterwards, we will focus on the subject/object ambiguity. The subject/object ambiguity in Basque caused by the homonymy of aboslutive plural and ergative singular (approximately 40% of the ambiguity left after morphosyntactic disambiguation). Functional ambiguity between subject and object is a widespread problem in Basque, where 22% of subjects and objects are ambiguous, and this ambiguity surfaces in 33% of the sentences. This problem is comparable to PP attachment ambiguities in other languages (Atutxa et al., 2012).

```
"<$.>"<PUNT_PUNT>"
"<Goizeko>"
  "goiz" ADJ C GEL S @<IZLG @IZLG>
  "goiz" ADJ C GEL S @OBJ @PRED @SUBJ
  "goiz" N C DIS ANIM- @ADLG
  "goiz" N C DES ANIM- @<IZLG @IZLG>
  "goiz" N C ABS ANIM- UNDET @OBJ @PRED @SUBJ
  "goiz" N C GEL ANIM- S @<IZLG @IZLG>
  "goiz" N C ABS ANIM- S @OBJ @PRED @SUBJ
"<bederatziak>"
  "bederatzi" DET ABS PL @OBJ @PRED @SUBJ
  "bederatzi " N NUMBER ABS PL @OBJ @PRED @SUBJ
  "bederatzi " N NUMBER ERG S @SUBJ
"<arte>"
  "artetu" V @NONFINITEV
  "arte"   N  C ANIM-  ABS @OBJ @PRED @SUBJ
  "arte"   N  C ANIM- @CM>
"<ez>"
"<nuen>"
 "*edun" VAUX NR_HURA NK_NIK @+JADLAG
 "*edun" VAUX REL NR_HURA NK_NIK @+JADLAG_MP_IZLG>
 "*edun" VAUX ZHG NR_HURA NK_NIK @+JADLAG_MP_OBJ
 "*edun" VAUX MOS NR_HURA NK_NIK @+JADLAG_MP_ADLG
 "ukan" V NR_HURA NK_NIK @+JADNAG
 "ukan" V REL NR_HURA NK_NIK @+JADNAG_MP_IZLG>
 "ukan" V ZHG NR_HURA NK_NIK @+JADNAG_MP_OBJ
 "ukan" V MOS NR_HURA NK_NIK @+JADNAG_MP_ADL
"<lortu>"
 "lortu" V PART @-JADNAG
 "lortu" V PART BURU  @-JADNAG
 "lortu" V PART ABS MG @-JADNAG_MP_OBJ
         @-JADNAG_MP_PRED @-JADNAG_MP_SUBJ
"<zure>"
"<ezpainak>"
 "ezpain"  N C ABS PL @OBJ @PRED @SUBJ
  "ezpain" N C ERG S @SUBJ
"<ikustea>"
  "ikusi"  V NOUNV KONP ABS @-JADNAG_MP_OBJ
           @-JADNAG_MP_SUBJ
  "ikusi"  N C ANIM- ABS S @OBJ @PRED @SUBJ

"<$.>"<PUNT_PUNT>"
```

Fig. 2: Morphological analysis

We will use the following sentence as a first example to illustrate the main steps of the methodology:

(1) *Goizeko bederatziak arte ez nuen lortu zure ezpainak ikustea* (stands for: Until nine o´clock in the morning I don´t achieve to see your lips).

In Figure 2 we have the morphological analysis of the sentence. We have simplified the analysis and we have concentrated on the phrases marked up in bold face *bederatziak arte* (stands for: until nine o´clock), which is the complex postposition structure that will be analyzed in further steps and *ezpainak* (stands for: lips). The ergative singular/absolutive plural ambiguity can be seen in the word *ezpainak* 'lips'. The form can potentially be a subject, object or predicate in absolutive case and a subject in ergative case. In this sentence it is an object and therefore in absolutive case. The ergative interpretation can be discarded based on valency requirements of the nominalized verb *ikustea* 'seeing'.

3.1 Lexical correction module

In the grammar for morphosyntactic disambiguation we have included a module of lexical correction for the treatment of complex postpositions. Prior to the morphosyntactic and syntactic disambiguation process, as a first step we design a lexical correction module composed by 155 SUBSTITUTE rules in order to eliminate unnecessary syntactic ambiguity. The complex postpositions have been processed at surface syntactic level instead of including these elements in the lexicon. As a result, we have inadequate syntactic tags for those structures.

Postpositions in Basque play a role similar to that of prepositions in languages like English or Spanish, so that, postposition suffixes are attached to the last element of the phrase. They are defined as "forms that represent grammatical relations among phrases appearing in a sentence" (Euskaltzaindia, 1994). We have treated at the surface syntactic level those postposition structures that are formed by a suffix followed by a lemma (postposition) that can be also inflected: *bederatziak arte* (stands for: until three).

The postposition element *arte* takes, as first component, an NP in absolutive case *–ak* (*bederatziak*, stands for nine). We can see that the postposition *arte* has four syntactic function tags corresponding to a noun and one for the non-finite verb interpretation. Besides, the first element of the postposition structure *bederatziak* has seven syntactic function tags taking into account the different morphological analysis. The elements of the complex postposition are tagged with the syntactic function tag corresponding to this structure. For instance, for the first element of

the postposition *bederatziak* the following SUBSTITUTE rule is applied:

```
SUBSTITUTE (@PRED @OBJ @SUBJ) (@CM>)
TARGET IZE-DET-IOR-ADJ-ELI-SIG
IF (0 ABS + MUGATUA) (1 POST-56IZE
+IZE_ABS_MG);
```

The SUBSTITUTE rule for the postposition *arte* is also based on the morphosyntactic information and in the previously defined postposition tagsets:

```
SUBSTITUTE (@PRED @OBJ @SUBJ) (@ADLG)
TARGET POSTPOSIZIOAK-5 IF (-1 IZE-DET-IOR-
ADJ-ELI-SIG + ABS + MUGATUA);
```

The main idea in both SUBSTITUTE rules is to substitute the syntactic function tag that is assigned by the morphological analyzer, because these syntactic function tags are not adequate for complex postpositions.

As a result of applying those rules we have reduced the starting syntactic ambiguity of the first example (see Figure 3):

```
"<bederatziak>"
  "bederatzi" N NUMBER ABS PL @CM>
"<arte>"
    "arte"   N  C ANIM- @ADVERBIAL
```

Fig. 3: Lexical correction of complex postpositions

The word *bederatziak* is holding after lexical correction rules the syntactic function tag @CM> (stands for: case-marker modifier) and *arte* the syntactic function tag @ADVERBIAL (stands for: adverbial). In the following step the grammar for disambiguating the syntactic function tags can select the appropriate function.

3.2 Reutilisation of chunker tags

When working with rich morphology languages like Basque it is crucial to make a basic distinction between disambiguation of main syntactic function tags and modifier syntactic function tags. The main syntactic function tags begin with the @-symbol (e.g. @SUBJ for subject, @OBJ for object, @PRED for predicate, etc.) The modifiers have tags that indicate the direction where the head of the phrase could be found but the modifiers and heads are not formally connected. The modifiers make use of the symbols < or > to indicate the direction in which could be found the head (e.g. @ND> for noun determiner, @NC> for noun complement, @CM> case-marker modifier, etc.)

The basic morphosyntactic disambiguation step deals with the disambiguation between main syntactic function tags and modifier syntactic tags, see figure 4:

```
"<Goizeko>"
  "goiz" N C ANIM- GEL S @IZLG>
```

Fig. 4: Basic morphosyntactic disambiguation

The word *goizeko* (stands for: in the morning) is a noun-complement and the head is on the right (that information is given by the symbol >).

On the output of the morphosyntactic disambiguation we applied one module of the rule-based chunker (RBC henceforth, Aranzabe et al., 2009), the one composed of 479 rules to deal with noun phrases. The chunker delimits the chunks with three tags, using a standard IOB marking style. The first one is to mark the beginning of the phrase (B-VP if it is a verb phrase and B-NP whether it's a noun phrase) and the other one to mark the continuation of the phrase (I-NP or I-VP, meaning that the word is inside an NP or VP). The last tag marks words that are outside a chunk.

In order to illustrate the reutilisation of the tags attached by the chunker, we will use the following example:

(2) *Microsoft etxea Word ari da euskaratzen* (stands for: Microsoft is translating Word into Basque).

After applying the rule-based chunker we get partial constituent analysis:

[3] **%SIH**[**Microsoft** etxea]**%SIB** [**Word**]**%SINT** [ari da euskaratzen]

We will focus on the NPs *Microsoft etxea* and *Word*. The NP *Microsoft etxea* is

[3] %SIH: initial part of a phrase; %SIB: ending part of a phrase and %SINT: phrase.

composed by the modifier *Microsoft* and the head *etxea.* In this case we base on the %SIH tag added by the chunker to discard the main syntactic function interpretations:

```
SELECT: (ZERO) IF (0C IZE LINK  0 (%SIH))
(1C (%SIB));
```

Regarding the NP *Word,* in this case it is constituted by one element, it is an independent phrase. We base on the %SINT tag to discard the modifier syntactic function tags:

```
REMOVE:kendu_zeroa, (ZERO) IF (0C
(%SINT));
```

3.3 Verb valency

There have been many attempts to include the subcategorization information in NLP parsers. Bick (2000) uses syntactic verb valency tags specifying e.g. transitivity and selection preferences for various NLP tasks. The use of verb valency is on a high level of grammatical analysis and requires a number of other linguistic resources. Bick (2000) uses tags specifying transitivity preferences such as <vt-vi>, meaning "preferably transitive, but potentially intransitive", but also selection preferences as in the entry for the Portuguese verb convidamos <+ACC-hum> 'invite', where the accusative needs to be a noun denoting a human.

Wiechetek and Arriola (2011) worked on applying verb valency information in the syntactic disambiguation process and they demonstrated that it is convenient.

We think that this kind of information should be included after some basic morphosyntactic disambiguation tasks have been done. It is clear that in some cases pure morphosyntactic information is not enough to solve the syntactic ambiguities. We have included by means of CG mapping rules the valency information developed in the work Building the Basque PropBank (Izaskun et al., 2010) into the CG grammar. In the previous version there was detailed subcategorization information for the most 100 used verbs.

We base on Aldezabal et al. (2010) converting this verb information into valency tags. Each verb can have several frames of argument constellations. In order to simplify the example we will take the verb *lortu* (stands for: to achieve, get), which has only one frame:

lortu V Agcase_ERG Agsyn_Subj Agsem_Human Thcase_Abs Thsyn_Obj

Arguments are ordered by semantic roles (e.g. agent, theme, topic, patient, location) because they are more unique than syntactic arguments (it is very common to have several adverbials in one sentence).

The semantic role level is furthermore perceived as being more abstract and therefore more language independent, which makes it suitable for reuse for other languages. Arguments have 3 possible attributes: case (or postposition) such as (nominative, accusative, ergative), syntactic function (subject, object, adverbial), and selection restrictions (human, concrete, place).

In the case of the verb lortu 'to achieve', the first argument, characterized by the semantic role agent, has the three attributes Agcase_ERG (ergative case) Agsyn_Subj (syntactic function subject) and Agsem_Human (selection restriction human).

The annotation of valency by means of Constraint Grammar rules has the following format adding the valency tags to the verb *lortu* 'achieve, get':

```
ADD (Agcase_Erg Agsyn_Subj Agsem_Human
Thcase_Abs Thsyn_Obj)

TARGET (ADI) IF (0 LORTU);
```

This format is sufficient for the annotation of a small amount of verbs for testing purposes. For a large-scale annotation of verbs we would like a verb database to be the basis from which tags are automatically induced.

Concerning the ambiguity of *ezpainak* (stands for: lips), the ergartive interpretation can be discarded based on valency requirements of the nominalized verb *ikustea* (to see). In this case we can´t make use of one important source of information for the disambiguation that is the agreement between the finite verb and the auxiliary.

When we deal with non-finite verbs, another strategy is to attach the verb auxiliary

information to non-finite verb forms in order to exploit this information when the auxiliary is elided. We make use of the following tags:

- DU: attached to those verbs that allow an auxiliary for transitive verbs, e.g. *lortu*.
- DA: attached to those verbs that allow an auxiliary for intransitive verbs, e.g. *loratu*.

However, in the illustrative example (1) that we have used for describing the methodology we can´t solve the syntactic ambiguity of *ezpainak* even with the auxiliary tag information.

Both morphosyntactic and syntactic ambiguity are tightly related. For that reason, in the first step we deal with morphosyntactic ambiguity and in the second step we deal with syntactic ambiguity. Our strategy to obtain the best disambiguation option was to do the morphosyntactic disambiguation first, and then once we have selected the absolutive or ergative case option we will deal with the syntactic ambiguity.

Another constraint grammar module contains disambiguation rules that make use of the valency. In the case of *lortu* 'achieve, get' in example (1), the ambiguity between the predicative and the object reading of *ezpainak* 'lips' is resolved by means of the valency of ikusi 'to see' and the object reading is selected by means of the following rule:

```
SELECT (@OBJ) IF (0 ABS LINK 0 OBJ) (NOT 0 ERG) (*1 Thcase_Abs LINK 0 Thsyn_Obj BARRIER ADI/ADL/ADT);
```

The other ambiguity in the sentence consists in the readings of the non-finite verbal noun *ikustea* 'seeing', which can be a subject, an object or a predicate. In order to select the object reading the rule checks if there is a verb, here *lortu* 'to achieve' to its left, that has an object in its valency:

```
SELECT (@-NON-FINITEVERB_CLAUSE_OBJ) IF (O NON-FINITE-VERB) (*-1 Thsyn_Obj BARRIER ADI/ADL/ADT);
```

4 Evaluation

The test corpus is divided into two parts: one for developing the grammar and the other one for testing the grammar on unseen corpus (53.324 tokens). We have tested the three basic experiments described in section 3. In the two first tests we have taken into account apart from POS and subcategory all the morphosyntactic information (case, number, type of subordinate clause, etc.). In the third one we tested the syntactic disambiguation and finally for verb valency we have used a smaller sample.

4.1 Lexical Correction module for Complex postpositions

We present firstly the figures of the morphosyntactic disambiguation grammar without the lexical correction module.

Words	**R**	**P**	**F**
Standard	91.40	71.92	80.50
Non-Standard	86.17	58.28	69.53
Out of Lexicon	81.06	36.08	49.93
Total	91.00	69.62	78.89
Total + PM	92.63	74.03	82.30

Table **1**: Initial results

(R=recall; P= precision; F= f-score; PM= puntuaction marks)

After applying the disambiguation grammar with the lexical correction module we obtain a moderate improvement of the results.

Words	R	P	F
Standard	92.42	72.58	81.19
Non-Standard	89.53	60.27	72.05
Out of Lexicon	77.80	36.45	49.64
Total	91.59	70.38	79.59
Total + PM	93.12	74.79	82.90

Table **2**: Lexical correction module effects

We have tested on a smaller sample (233 words) that doesn´t contain non-standard words. Applying the same grammar we obtain better results, recall 94.93 and precision 76.78.

These results show that the effect of the non-standard words on the disambiguation of the standard words should be taken into account when designing the grammar.

4.2 Chunker tags reutilisation

Testing at morphosyntactic level the effect of combining the morphosyntactic disambiguation grammar and the information of the chunker we don´t get better results.

Words	R	P	F
Standard	91.42	73.49	81.48
Non-Standard	88.66	61.00	72.27
Out of Lexicon	72.67	37.85	49.78
Total	90.73	71.47	79.96
Total + PM	92.41	75.72	83.24

Table **3**: Chunker tags reutilisation effects

In the next section we will show the effects on the disambiguation of syntactic function tags.

When analyzing the results shown in Table 2 and Table 3 we should take into account some features of the disambiguation process:

- Complexity of some ambiguities: case (need to deal with the agreement of verbs with subject, object, etc.); the type of subordinate sentence in auxiliary verbs; elided or non-elided element in auxiliary verbs; relative clause versus past tense verb; etc.
- Treatment of variants
- Some errors of the chunker
- Some divergences with the Gold Standard

4.3 Combining lexical correction module and chunker tags

The syntactic ambiguity rate of the testing sample is 5.469 syntactic tags per word. We have tested the effect of combining different modules.

	Input	Output1	Output2	Output3
Anal./Token	5.469	1.930	1.260	1.217
Error rate		4.17	8.6	9.2

Table **4**: Syntactic disambiguation results

The first results (Output1) have been obtained applying just the morphosyntactic disambiguation grammar, the following results for the second output (Output2) have been obtained by means of the disambiguation grammar for syntactic functions and the third one (Output3) has been obtained applying the grammars in this way: first the morphosyntactic disambiguation grammar, secondly on the output of the morphosyntactic grammar the chunker and finally on the output of the chunker the syntactic function disambiguation grammar.

In Table 4 we can see that as the CG-based syntactic disambiguation grammars and the chunker are applied after morphological processing, the errors are propagated and augmented.

4.4 Verb valency

In many cases, we have seen that pure syntactic information is not sufficient for the morpho-syntactic disambiguation of nouns, and richer linguistic information is needed. For instance, in example (1) we have seen that we need verb valency information to solve the ambiguity of *ezpainak*.

The test cases used in this experiment regard morphosyntactic disambiguation; especially we will focus on the ambiguity of the suffix *–ak*. The test includes a set of 10 verbs for disambiguation. In the text that we have chosen for the experiment there are 177 different verbs, so the ten verbs that we have studied represent the 5,6% of the verbs of the sample. This should be considered when we are talking about the effect of the valency information in the syntactic disambiguation process.

Due to the low coverage, they were tested on a corpus containing sentences including the annotated verbs rather than running text.

Wrong applications of rules are mainly due to the occurrence of several verbs with different valencies in one sentence and scope mistakes of the rules. Another reason is low coverage of semantically annotated nouns. The rules involved in the disambiguation of the absolutive-ergative syncretism correctly solve the ambiguity in 64.2% of the cases and incorrectly in 28.5% of the cases, 7% of the cases are left yet ambiguous. The errors are due to the following main reasons:

- Concerning the verb auxiliary tag and the strategy for checking the agreement between the case, the verb and the auxiliary, the main problem is caused by the ellipsis of verbs. This phenomenon is observed in coordination structures and comparatives
- Concerning the valency information: we have on the one hand that the rule disambiguates based on the valency information of an unrelated verb, and on the other hand that the semantic information of the nouns is missing. In order to improve the results generalizing and extending the subcategorization information to more verbs, refining the disambiguation rules based on verb subcategorization and finally improving the semantic noun sets to meet the lexical selection restrictions of the verbs will be necessary.

For a thorough evaluation, the resources need to be improved, first a gold standard of the surface oriented syntax and the implementation of the utility for deriving automatically those correct analysis that have been removed by the CG3 rules. When those are available, a thorough evaluation of more verbs and syntactic disambiguation is planned.

5 Conclusions

This paper has described the work for modeling the surface syntactic disambiguation module reusing and establishing the application order of some of the existing modules in the general framework for the surface syntactic processing. We have started out by setting up the basic steps that should be done in order to improve the syntactic disambiguation module in order to achieve a robust disambiguation as a basic step for further syntactic processes of rich morphology languages like Basque. The results show a modest improvement, although they also present interesting lines for further research. We plan to go further with new experiments based on the combination of the different modules. Furthermore, we would like to apply machine learning-based techniques following the way suggested by Bick (2013) to optimize our grammars.

Finally, future work would certainly profit from access to a larger and revised Gold Standard, to investigate the divergencies on the analyses and to establish the general criteria for solving those differences. In fact our Gold Standard is under development and needs an exaustive study from a qualitative point of view in order to clarify the different sources of error observed when testing our grammars. We should analyze which errors are caused by the grammar and which errors are due to the inconsistencies and incorrect tags in the Gold Standard.

Acknowledgments

This research was supported by the Department of Industry of the Basque Government (IT344-10, S PE11UN114), the University of the Basque Country (GIU09/19).

References

Abney S. P. 1997. *Part-of-speech tagging and partial parsing*. S. Young and G. Bloothooft, editors, Corpus-Based Methods in Language and Speech Processing, Kluwer, Dordrecht.

Aduriz, I., Aldezabal, I., Alegria, I., Arriola, J., Díaz de Ilarraza, A., Ezeiza, N., Gojenola, K. 2003. Finite State Applications for Basque, Workshop on Finite-State Methods in Natural Language Processing. 11th Conference of the European Chapter of the Association for Computational Linguistics, Budapest, Hungary 2003, pp. 3-11.

Agirre E., Atutxa A., Sarasola K. 2012. Contribution of Complex Lexical Information to Solve Syntactic Ambiguity in Basque. Proceedings of COLING 2012, pp: 97—114.

Aldezabal, I., Alegria, I., Ansa O., Arriola, J., Ezeiza, N. 1999. Designing spelling correctors for inflected languages using lexical transducers, Proceedings of European Chapter of the Association for Computational Linguistics, Bergen, Norway, June 1999, pp. 265-266.

Aldezabal, I., Ansa, O., Arrieta, B., Artola, X., Ezeiza, A., Hernández, G., Lersundi, M. 2001. EDBL: a General Lexical Basis for the Automatic Processing of Basque, Proceedings of the IRCS Workshop on Linguistic Databases. Philadelphia (USA), December 2001, pp. 1-10.

Aldezabal, Izaskun, María Jesús Aranzabe, Arantza Díaz de Illarraza, Ainara Estarrona and Larraitz Uria. 2010. 'Euspropbank: Integrating semantic information in the basque dependency treebank', Computational Linguistics and Intelligent Text Processing pp. 60–73.

Alegria, I., Artola, X., Sarasola, K., Urkia, M. 1996. Automatic morphological analysis of Basque. Literary & Linguistic Computing, 11: 193-203.

Alegria, I., Artola, X., Sarasola, K. 1997. Improving a Robust Morphological Analyser using Lexical Transducers. In Mitkov, R. and Nicolov, N. (eds), Recent Advances in Natural Language Processing. Current Issues in Linguistic Theory (CILT) series, 136. John Benjamins publisher company, pp. 97-110.

Maria Jesus Aranzabe, Jose Maria Arriola and Arantza Díaz de Ilarraza. 2009. Theoretical and Methodological issues of tagging Noun Phrase Structures following Dependency Grammar Formalism. In Artiagoitia, X. and Lakarra J.A. (eds) Gramatika Jaietan. Patxi Goenagaren omenez. Donostia: Gipuzkoako Foru Aldundia-UPV/EHU.

Bick, E. 2000. *The Parsing System 'Palavras': Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework*, Aarhus University Press, Aarhus.

Bick, Eckhard. 2013. ML-Tuned Constraint Grammars. In: Proceedings of the 27th Pacific Asia Conference on Language, Information and Computation, pp. 440-449. Taipei: Department of English, National Chengchi University.

Didriksen T. 2010. *Constraint Grammar Manual: 3rd version of the CG formalism variant*. Grammar-Soft Aps, Denmark.

DoleŽalova, J. and PetkeviČ, V. 2007. Shallow parsing of Czech sentence based on correct morphological disambiguation. I: Kosta, P. and Schürcks, L. (eds.), Linguistic Investigations into Formal Descriptionof Slavic Languages. Contributions of the Sixth European Conference (FDSL-6), Peter Lang:Frankfurt am Main, 53-64.

Euskaltzaindia. 1994. *Basque Grammar: First Steps (in Basque)*. Euskaltzaindia.

Karlsson, Fred, Atro Voutilainen, Juha Heikkilä & Arto Anttila. 1995. Constraint grammar: A language-independent system for parsing unrestricted text. In Natural Language Processing, No 4. Berlin and New York: Mouton de Gruyter.

Laka, I. 1996. *A Brief Grammar of Euskara, the Basque Language*, Euskararako Errektoreordetza, EHU.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kubler, S., Marinov, S., and Marsi, E. 2007. Maltparser: A language-independent system for data-driven dependency parsing. Natural Language Engineering, 13(2):95.

Samuelsson C. and Voutilainen A. 1997. Comparing a linguistic and a stochastic tagger. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the Eighth Conference of the European Chapter of the Association for Computational Linguistics. ACL.

Tapanainen, Pasi. 1996. *The Constraint Grammar Parser CG-2*. University of Helsinki Publications No. 27.

Tapanainen P. and Järvinen T. 1997. A non-projective dependency parser. In Proceedings of the Fifth Conference on Applied Natural Language Processing ACL.

Wiechetek Linda, Arriola J.M. 2011. An Experiment of Use and Reuse of Verb Valency in Morphosyntactic Disambiguation and Machine Translation for Euskara and North Sámi In: Constraint Grammar Workshop at NoDaLiDa 2011.

Optimizing the Oslo-Bergen Tagger

Eckhard Bick
University of Southern Denmark
Odense
eckhard.bick@mail.dk

Kristin Hagen & Anders Nøklestad
University of Oslo, Norway
kristin.hagen@iln.uio.no
anders.noklestad@iln.uio.no

Abstract

In this paper we discuss and evaluate machine learning-based optimization of a Constraint Grammar for Norwegian Bokmål (OBT). The original linguist-written rules are reiteratively re-ordered, re-sectioned and systematically modified based on their performance on a hand-annotated training corpus. We discuss the interplay of various parameters and propose a new method, continuous sectionizing. For the best evaluated parameter constellation, part-of-speech F-score improvement was 0.31 percentage points for the first pass in a 5-fold cross evaluation, and over 1 percentage point in highly iterated runs with continuous resectioning.

1 Introduction and prior research

Typical Constraint Grammars consist of thousands of hand-written linguistic rules that contextually add, change or discard token-based grammatical tags for lemma, part-of-speech (POS), morphological feature-value pairs, syntactic function, semantic roles etc. Each rule interacts intricately with all other rules, because the application of a rule will change the grammatical sentence context for all subsequent rules, and section-based rule iteration further complicates this process. Thus, a CG grammarian can only trace rule effects for one token at a time, and only for rules that actually are used. As a consequence, improvements are made in a piecemeal fashion, while it is practically impossible for a human to meaningfully rearrange the grammar as a whole. We therefore believe that most CGs could potentially profit from data-driven, automatic optimization.

Early work in this direction was the µ-TBL system (Lager 1999), a transformation based learner that could be seeded with CG rule templates, for which it would find optimal variations and rule order with the help of a training corpus. However, µ-TBL did not perform as well as human grammars, and could only handle n-gram-type context conditions. Lindberg & Eineborg' Progol system (1998) induced CG REMOVE rules from annotated Swedish data and achieved a recall of 98%, albeit with a low precision (13% spurious readings). The first system to use automatic optimization on existing, linguist-written grammars, was described in Bick (2013), and achieved a 7% error reduction for the POS module of the Danish DanGram[1] parser. Results were twice as good for a randomly reduced grammar with only 50% of the original rules, indicating a potential for grammar boot-strapping and grammar porting to another language or genre, where only part of the existing rules would be relevant, which was later shown to be true for at least the Danish-English language pair (Bick 2014). In the work presented here we examine how Bick's optimization method holds up for a Norwegian Bokmål CG, and discuss different parameter options.

2 The Oslo-Bergen Tagger (OBT)

The Oslo-Bergen Tagger is a rule-based Constraint Grammar (CG) tagger for the Norwegian varieties Bokmål and Nynorsk. Below we will give a brief presentation of the OBT history, the architecture behind it, and the

[1] DanGram is accessible on-line at http://visl.sdu.dk/visl/da/parsing/automatic/parse.php

OBT performance. The presentation will focus on the Bokmål tagger.

2.1 History

OBT was developed in 1996–1998 by the Tagger Project at the University of Oslo. The linguistic rules were written at the Text Laboratory in the CG1 rule framework (Karlsson et. al. 1995). Originally, the tagger used a rule interpreter from the Finnish company Lingsoft AB. The tagger performed both morphological and syntactic analysis (Johannessen et. al. 2000). In 2000 the preprocessor and the rule interpreter was replaced by a reimplementation in Allegro Common Lisp made by Aksis (now Uni Research Computing) in Bergen, and the tagger was named The Oslo-Bergen Tagger.

Within the project Norwegian Newspaper Corpus (2007-2009), OBT was once again converted. The CG rules were semi-automatically transformed from CG1 to the new CG3 formalism (Bick & Didriksen 2015), and Uni Research Computing made a stand-alone version of the preprocessor that could work together with the CG3 compiler[2].

Finally, a statistical module was trained to remove the last ambiguity left by OBT. This module also performed lemma disambiguation, a task the original OBT did not do. The new system was called OBT+stat and is described in more detail in Johannessen et. al. (2012).

In this article we will focus on the morphological CG part of OBT+stat since the optimization is performed on the morphological rules without regard to the statistical module.

2.2 The architecture behind OBT

The morphological part of OBT consists of two modules:

a) Preprocessor: The preprocessor is a combined tokenizer, morphological analyzer and guesser that segments the input text into sentences and tokenizes their content. There are special rules for names and various kinds of abbreviations. Each token is assigned all possible tags and lemmas from the electronic lexicon *Norsk ordbank* (Norwegian Word Bank). The Bokmål part of this lexicon contains more than 150 000 lemmas together with inflected forms (Hagen & Nøklestad 2010). The guesser includes a compound word analyzer and manages productively formed compounds as well as unknown words (Johannessen & Hauglin 1998).

b) Morphological disambiguator: The morphological disambiguator is based on CG3 rules that select or remove tags attached to the input tokens. There are 2279 linguistic rules in this module. 693 of them are rules for specific word forms. 1371 are SELECT rules and 908 are REMOVE rules.

The tag set is rather large, consisting of 358 morphological tags. The part of speech classifications, which include information about morphosyntactic features, are performed in accordance with Norsk referansegrammatikk (Faarlund et. al. 1997).

2.3 OBT performance

The original tagger was tested on an unseen evaluation corpus of 30,000 words taken from a wide variety of material such as literary fiction, magazines and newspapers. The recall and precision were 99.0 and 95.4 percent respectively, with a combined F-measure of 97.2 (Hagen & Johannessen 2003:90)

After the conversion to CG3 format, recall remained at 99.0 percent while precision increased to 96.0 percent, resulting in an F-measure of 97.5 (see the OBT homepage).

3 Grammar optimization

For our experiments we used a 155.000 word[3] corpus with hand-corrected OBT tags covering POS and inflection. The original corpus was divided into a (larger) development section and a (smaller) testing section. We used this division for the parameter-tuning experiments, but for the final results, in order to avoid a bias from human rule development, we fused these sections and created sections of equal size to be used in 5-fold cross evaluation. We also adapted the OBT grammar itself, because its rules have operators, targets and contexts on separate lines, sometimes with interspersed comments. Although CG3-compatible, this format had to be changed into 1-line-per-rule in order to make rule movements and rule ordering possible.

3.1 Optimization technique and parameters

The optimization engine works by computing for

[2] CG3 (or vislcg-3) is open source and available at SDU: http://visl.sdu.dk/constraint_grammar.html

[3] when counting all tokens, including punctuation

each rule in the grammar a performance value based upon how often it selects or removes a correct reading in the training corpus. Rules are then reordered, removed or modified in reaction to this value, and the effect measured on the test corpus. After this, the process is repeated with the new grammar, and so on. As in Bick (2013, 2014), we investigated the following actions and parameters:

(1) Sorting rules according to performance

(2) Promoting good rules the next-higher section (good = error percentage lower than T/2)

(3) "Demoting" bad rule to the next-lower section (bad = error percentage higher than T)

(4) "Killing" very bad rules (error percentage is over 0.5, doing more bad than good)

(5) Add relaxed versions of good rules, by removing C- (unambiguity-) conditions and, conversely, changing BARRIERs into CBARRIERS

(6) Replace bad rules with stricter versions, by adding C conditions and turning CBARRIER into BARRIER.

(7) "Wordform stripping" - i.e. adding a relaxed version of a wordform-conditioned rule without this condition, thus allowing it to work on all tokens.

Performance values (recall, precision and F-score) for the modified grammars, in the tables below, are for POS (i.e. without inflection), and because OBT has more than one tag for a comma, evaluation also includes punctuation.

3.2 What did not work, or in limited ways

The maybe most obvious step, sorting rules section-internally after each iteration, decreased performance. Sorting was only helpful in dry-run[4] mode, for an initial ordering of the original grammar and after the 1. iteration's addition of modified rules[5], and only in the combination of complete sorting plus resectioning. regardless if sorting was performed for sections individually or for the whole grammar together. The importance of re-sectioning indicates that the existence and placement of sections is an important parameter, that the concept of a good/bad rule is section-dependent[6], and that it is an important optimization parameter. All runs in the table below were with standard PDK and 1-time dry-run sorting and examine different combinations of iterative sorting. As can be seen, section-internal sorting worked worst (F=96.26), having a kill-section helped more (F=96.49) than factoring in human sectioning as a weight (F=96.26). But in all instances, iterated sorting was worse than 1-time sorting (F=96.56). Increasing the number of sections to 11 led to a certain recovery of F-scores in hight iterations, but could not beat the first run in our experiment window (50 runs).

PDK	ite-ration	Recall (%)	Precision (%)	F-score
original grammar		98.08	94.27	96.13
no iterated sorting	3	98.72	94.50	96.56
sorting 5 sections	1	98.23	94.66	96.41
sorting 11 sections (--> F-score recovery)	1	98.22	94.66	96.41
sorting11, +kill	1	98.33	94.72	96.49
sorting2, +kill	1	98.32	94.72	96.49
sorting11, -kill, human section-weighting	11	98.25	94.36	96.26
sorting5 sect.-internal	14	96.99	94.35	95.65

Effects of iterated sorting

Word form stripping was reported in Bick (2014) to have had a positive effect in cross language grammar porting, but we could not reproduce this effect in the monolingual setup with the same parameters (PDK, 1 dry-sorting). Cross-language, the method can possibly offset the problem that wordforms from one language don't exist in the other, creating versions of the rules

[4] In a dry-run call, CG3 applies all rules once, but without making changes to the input. Rule tracing in a dry-run will therefore is a way to measure how rules would perform in isolation without actually running thousands of 1-rule minigrammars.

[5] The rule change that profited from sorting was stripping of wordform target conditions, because this change creates very unrestrained and dangerous rules.

[6] 10% errors, for instance, is good in a heuristic section, where most of the disambiguation has been done already, but may be bad if the rule is run too early, where the error rate may be the same in relative terms, but worse in absolute terms, because it will apply to more cases.

that work at least in terms of POS etc., while this effect is not relevant monolingually, were wordform rules just get more risky by losing their pivotal wordform condition. However, with a subsequent second dry-run sorting and very high, section-growing iteration counts, the optimizer seems to be able to identify the useful subset of wordform-stripped rules and find acceptable section placements for them (cf ch. 4).

In terms of numerical parameters we experimented with the error threshold, but failed to find a better value the 0.25 suggested by (Bick 2013) - both lower and higher thresholds decreased performance, independently of other parameter settings[7].

3.3 What did work

While the negative effect of sorting confirms results for Danish, there was a surprising difference regarding promoting, demoting and killing. For DanGram, demoting and killing rules had a beneficial effect, while promoting rules had almost no effect. For OBT, however, rule promotion (P) was important, and for a combination of only demoting (D) pseudokilling (k3) extra iterations did not yield a better effect than the initial one-time sectioned rule-sorting (S0).

	R	dR	P	dP	F	dF
unaltered grammar	98.08		94.27		96.13	
S0D(s)k3, -w (i=1/50[8])	98.22	0.14	94.66	0.39	94.41	0.28
S0PDsk3, -w (i=45/50)	98.71	0.63	94.85	0.58	96.75	0.62

Effect of rule promoting

This can possibly be explained by different grammar properties: OBT is recall-optimized (there are about 4 times as many spurious readings than errors[9]), while DanGram resolves almost all ambiguity (i.e. one reading per token), so that precision will roughly equal recall. Therefore OBT profits from promoting good rules (so they can do more disambiguation work), while DanGram, on the recall side, profits from demoting and killing rules (preventing them from removing correct reading). Another difference between the two grammar is that OBT has a higher proportion of SELECT rules and a lower proportion of C contexts and BARRIERs. Because C and BARRIER contexts are harder to instantiate[10] (needing more supporting context disambiguation first), and because SELECT resolves ambiguity in one go that REMOVE rules would have needed several steps for, it can be said that DanGram's rules work more incrementally and indirectly, while OBT is more direct in its disambiguation. This harmonizes with the finding that promoting helped OBT, but not DanGram, because promoting makes sense for rules that are formulated as "absolute truths" (SELECT rules), but doesn't help for rules whose C and BARRIER contexts force them to wait for other rules to work first anyway.

	DanGram	OBT
morph. rules	5120	2215
REMOVE	2837 (55.4%)	898 (40.5%)
SELECT	2178 **(42.5%)**	1312 **(59.2%)**
OTHER	105 (2.1%)	5 (0.2%)
wordform rules	1808 (35.3%)	777 (35.1%)
contexts	17539 (3.48/rule)	11525 (5.55/rule)
C conditions	4549 **(25.9%)**	1303 **(11.3%)**
NOT	3239 (18.5%)	5007(43.4%)
NEGATE	477 (2.7%)	5 (0.0%)
LINK	4192 (23.9%)	2705 (23.5%)
BARRIERS	3554 **(20.3%)**	927 **(8.0%)**
CBARRIERS	276 (1.6%)	44 (0.4%)
global contexts	4597 (26.2%)	3326 (28.9%)

Grammar properties DanGram - OBT

[7] However, we did not have computational resources to experiment with exponent changes or the *0.5 difference between good rule and bad rule thresholds.

[8] F-Score stabilized below the initial sorting optimization, independently of whether new sections were added or not, for the latter from iteration 11, at 96.224, for the former from iteration 5, at 96.222.

[9] This 1:4 ratio holds for both for our own tests (R=98, P=92) and Hagen & Johannessen's (2003) evaluation (R=99, P=96)

[10] The effect may be somewhat compensated for, however, by the higher percentage of NOT rules in OBT, which also makes rules more "cautious".

Killing rules does have a slight initial effect in OBT, but over several iterations, the effect is negligible. We therefore introduced a compromise parameter into the optimizer, defining "killing" as moving a rule 10[11] sections down, rather than removing it completely it from the grammar. This did not have an adverse effect, and while most of these rules never moved up again in later iterations, a few of them can do a little late heuristic work in low sections. The method also preserves the rules in question for use with other text types, reducing the risk of over-fitting a very lean grammar to a specific training corpus. Metaphorically speaking, we preserve a varied "gene pool" of rules as a reserve for a new data environment. In the same vein we decided to preserve unused rules (rather than going for an efficiency gain in a leaner grammar)[12].

We also noted a positive effect, especially on precision, from adding relaxed versions of good rules and a smaller positive effect from making bad rules stricter.

3.4 Sectionizing

Sectionizing the grammar is not only a key parameter with any sorting configuration, but also in general. The standard optimizer from (Bick 2013) adds one new first section for moving up rules from the original first section, and had a positive effect from this. But we wanted to test the hypothesis that more sections will lead to a more fine-grained quality ordering of rules and exploit the fact that the CG compiler will try rules *twice* (or even three times) within the same section, and rerun higher-section rules before it starts on the next lower (= more heuristic) section. So we added new top and bottom sections at each iteration, allowing the grammar to differentiate more when promoting and demoting existing and changed rules (PDN).

	R	dR	P	dP	F	dF
	98.08		94.27		96.13	
S0PDk3, -w (i=1)	98.22	0.14	94.66	0.39	94.41	0.28
S0PDNk3, -w (i=45/50)	98.71	0.63	94.85	0.58	96.75	0.62

Effect of sectionizing (S0=1 dry sorting, k3=killing by moving 3 sections down)

Continually adding extra sections to the grammar had a marked dampening effect on the performance oscillation of the iteration curve. Also, performance kept increasing, with late stabilization, and a maximum at iteration 45 in the example, whereas most runs with a stable section number had their F-Score maximum already in the first iteration, and stabilized somewhere between the unoptimized performance and this first maximum. In theory, the section-adding technique can end up section-separating individual rules, making the process equivalent to precise one-by-one rule ordering, which conceptually beats the group ordering achieved by fixed-section optimization.

The obvious price for adding new sections was slower execution - with a worst case ceiling at quadratic growth in time consumption (because the compiler reruns lower sections before embarking on a new one). In practice, however, the distribution of rules across sections was lumpy. For instance, when adding sections for top/bottom-moved rules but not removing empty sections, the grammar from iteration 51 in the above test had 20 used and 32 empty sections, grammar 100 had 15 used and 62 empty sections[13].

It should be noted that once a grammar is optimized, execution time can be improved at a fairly small price by reducing the number of sections. Thus, F-score decreased only marginally when we resectioned an optimized 50-section grammar to 6 equal sections. However, the effect on recall and precision was unequal - the former fell by 0.5 percentage points, the latter rose by 0.4 percentage points. Both effects can be explained by strong but dangerous rules acting too early. Still, in average F-score terms, optimization with a very fine-grained section skeleton will more or less amount to individual rule ordering and therefore

[11] We also tried a lower number, 3, which did not work as well.

[12] Rules may be unused only because they are placed in a certain section, so they can also come back into play in later iterations during training, when other - higher - rules, that did their work for them, are demoted to a section below a given inactive rule.

[13] In a final, working grammar, empty sections should of course be removed, but during optimization empty in-between sections allow more fine-grained rule differentiation and seemed to have a slight positive effect

tolerate de-sectioning. In our experiment, even removing *all* sections borders still did not seriously harm F-Score. Thus, sections are important for the optimization process, but in an optimized grammar they are less important than in the human original.

4 Final results

To achieve as reliable results as possible, we used a 5-fold cross evaluation for the final evaluation. For the best parameter setting (dry sorting, promoting and demoting with new sections, pseudokilling), the initial F-score optimization gain (1. iteration) varied from 0.27 to 0.36 percentage points between the 5 combinations, with an absolute F-score spread of 95.58 to 96.60. As might be expected, the weakest sections (3 and 4) profited the most from optimization.

S0PDNk3, -w	R	dR	P	dP	F	dF
1: (i=1/6)	98.63	0.16	94.71	0.43	96.60	0.30
2: (i=1/6)	98.42	0.20	94.95	0.35	95.65	0.28
3: (i=1/6)	97.67	0.22	93.58	0.48	95.58	0.36
4: (i=1/6)	97.38	0.26	93.9	0.44	95.60	0.36
5: (i=1/6)	98.24	0.16	94.7	0.38	96.41	0.27
average (i=1/6)	98.07	0.20	94.35	0.42	96.17	0.31

Table: Performance spread across the corpus

(S0=dry run sorting only (with resectioning=5 and 10-killing), PDs=Promoting & Demoting with new sections per iteration, k3=3-section demoting instead of killing, -w=no wordform stripping)

With new promoting/demoting sections for every round, performance maxima tend to occur late in the iteration cycle, so to investigate the ultimate improvement potential, we used the section with the weakest dry run (3) and let iteration run for 100 rounds. Because each such run took over half a day on our hardware, we were only able to investigate few parameter settings at the time of writing. The best result, an F-Score improvement (dF) of 1.31, was achieved with reintroducing ordinary killing at iteration 15, and a maximum in round 86. When introducing a second "dry" sorting in iteration 2, i.e. after the addition of relaxed and stricted rules, and ordinary killing from iteration 4, much shorter training runs were needed (with an asymptotic maximum already in round 16), albeit at a slightly lower level (dF=1.09). With this setting, even word form-stripping could be tolerated, with a maximal dF of 0.51 in round 63. Here, too, the growth curve was asymptotic, but it still oscillated until the end, so later maxima can't be entirely ruled out - and would make sense, given the very "un-cautious" character of wordform-stripped rules. It is probably these "un-cautious" rules that explain why wordform-stripped rules benefited precision twice as much as recall, while high iterations otherwise had a strong recall bias.

Performance oscillations for the training corpus correlated with performance on the test set, but test corpus results for grammars with training corpus maxima[14] could deviate up to 0.1% from the actual test corpus peak, which is a rough measure for the expected "performance impredictability" when using ML-optimized grammars on unknown data.

test chunk 3[15]	R	dR	P	dP	F	dF
i=0	97.46		93.10		95.23	
S0PDNk3K15, -w (i-max=86/100)	99.09	1.63	94.11	1.01	96.54	1.31
training i=86	99.26		84.99		91.57	
S2PDNk3K4, -w (imax = 16 const.)	98.86	1.40	93.90	0.80	96.32	1.09
training i=16	99.07		85.41		91.73	
S2PDNk3K4, -w (i-max=63/100)	97.80	0.34	93.76	0.66	95.74	0.51
training i=63	98.04		89.18		93.40	

Best case scenario - "Unlimited" iterations

5 Conclusion

We have demonstrated that ML-optimization can be successfully performed for a Norwegian constraint grammar, and explored a new sectioning strategy and the respective influences

[14] F-scores for training runs appear to be lower than for the test corpus, but only because the optimizer in the training runs evaluates against full tag lines, i.e. with inflection and secondary tags, not just POS.

[15] We ran the second parameter setting for the original training/test-split too, with the same asymptotic result. F-Sore topped at 96.67, 0.53 percentage point above the unaltered grammar (F=96.14), but the lower increase has to be interpreted on the basis of a higher base line.

of rule sorting and rule movements. For most parameter constellations, repetition of optimization runs did not lead to a better performance than a single pass, unless each iteration is allowed to add new sections. The first-pass average improvement in 5-fold cross evaluation was 0.31 percentage points (F-Score 96.17), similar to Danish results reported in Bick (2013), but with added sectionizing and long iterations, improvements of over 1 percentage point were seen, corresponding to a 30% improvement in relative terms. The immediate effect was best for precision, but with high iterations, recall was affected most, with a 60% improvement in relative terms.

Future work would certainly profit from access to a large computer cluster, to investigate the millions of possible combinations of incremental parameter changes. Also, it would be interesting to get the human linguist back into the loop, to see if some of the rules slated for killing or demotion by the optimizer can be "saved" by additional context conditions instead, and if the best selected generalized variants of wordform rules can be used for further development.

References

Bick, Eckhard. 2013. ML-Tuned Constraint Grammars. In: Proceedings of the 27th Pacific Asia Conference on Language, Information and Computation, pp. 440-449. Taipei: Department of English, National Chengchi University.

Bick, Eckhard. 2014. ML-Optimization of Ported Constraint Grammars. In: Calzolari, Nicoletta et al. (eds.), Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC2014 (Reykjavik, May 28-30, 2014). pp. 3382-3386.

Bick, Eckhard & Didriksen, Tino. 2015. CG-3 - Beyond Classical Constraint Grammar. In: Proceedings of NoDaLiDa 2015 (forthcoming).

Faarlund, Jan Terje, Lie, Svein & Vannebo, Kjell Ivar. 1995. Norsk referansegrammatikk. Oslo: Universitetsforlaget.

Hagen, Kristin & Nøklestad, Anders. 2010. Bruk av et norsk leksikon til tagging og andre språkteknologiske formål. *LexicoNordica* 2010 (17) pp. 55-72.

Hagen, Kristin & Johannessen, Janne Bondi. 2003. Parsing Nordic Languages (PaNoLa) - norsk versjon. Nordisk Sprogteknologi 2002. Museum Tusculanums Forlag, Københavns universitet.

Johannessen, Janne Bondi and Helge Hauglin. 1998. An Automatic Analysis of Norwegian Compounds. In Haukioja, T. (ed.): Papers from the 16th Scandinavian Conference of Linguistics, Turku/Åbo, Finland 1996 : 209-220.

Johannessen, Janne Bondi, Hagen, Kristin & Nøklestad, Anders. 2000. A Constraint-based Tagger for Norwegian. In 17th Scandinavian Conference of Linguistics [Odense Working Papers in Language and Communication 19].

Johannessen, Janne Bond; Hagen, Kristin; Lynum, André; Nøklestad, Anders. 2012. OBT+stat: A combined rule-based and statistical tagger. In Andersen, Gisle (ed.). Exploring Newspaper Language: Using the web to create and investigate a large corpus of modern Norwegian, s. 51–66.

Karlsson, Fred, Voutilainen, Atro, Heikkilä, Juha & Anttila, Arto. 1995. Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text. In Natural Language Processing, No 4. Berlin and New York: Mouton de Gruyter.

Lager, Torbjörn. 1999. The µ-TBL System: Logic Programming Tools for Transformation-Based Learning. In: Proceedings of CoNLL'99, Bergen.

Lindberg, Nikolaj & Eineborg, Martin. 1998. Learning Constraint Grammar-style Disambiguation Rules using Inductive Logic Programming. COLING-ACL 1998: 775-779

Norsk Ordbank 'Norwegian Word Bank'. 2010. http://www.hf.uio.no/iln/om/organisasjon/edd/forsking/norsk-ordbank/.

Oslo-Bergen Tagger homepage. <http://tekstlab.uio.no/obt-ny/>.

A Constraint Grammar POS-Tagger for Tibetan

Edward Garrett
SOAS, University of London
eg15@soas.ac.uk

Nathan W. Hill
SOAS, University of London
nh36@soas.ac.uk

Abstract

This paper describes a rule-based part-of-speech tagger for Tibetan, implemented in Constraint Grammar and with rules operating over sequences of syllables rather than words.

1 A POS-tagger for Tibetan

In earlier work, we described a rule-based tagger for Classical Tibetan, implemented using regular expressions (Garrett et al., 2014). Since then, the rule tagger has kept moving: our grammatical understanding of Tibetan has evolved by reading and hand tagging 236,167 Tibetan words.[1]

The primary purpose of the rule-based tagger has been to speed up the hand tagging of texts. The output of a lexical tagger, assigning to each word all of its possible tags, is fed into the rule tagger, which then removes only those analyses precluded by the context. The result of this process is highly ambiguous, with an average of 1.3936 remaining tags per word. However, it is also highly accurate, with 99.8 percent of words receiving the correct tag (van Halteren, 1999). In consequence, the human can focus their efforts on determining the correct tag for words that remain ambiguous, without needing to worry about words that the rule tagger is sure about.

In the development of the tagger, no attempt was made to divide the corpus into separate training and test sets. To do so would have been counterproductive, as it would have required us to read and tag texts without learning from them. To the contrary, we have seized every chance to develop and further refine the rule set. It is a pleasing result and some measure of success that the tagger performs well when evaluated against the materials that inspired it.

Despite its initial promise and usefulness, the regular expressions tagger has been deprecated. It soon became evident that maintaining the rule set required a regular expressions wizard with a keen eye for slashes. Being both difficult to read and difficult to maintain, the rule set seemed an unlikely candidate for linguists to build on and continue to use in the future. An additional purely hypothetical concern was that the tagger might soon require rules that would exceed the expressive capacity of regular expressions.

These concerns, combined with the fortuitous discovery of a new framework, led us to translate the entire rule set into CG-3 (Bick & Didriksen, 2015), the latest version of Constraint Grammar. We translated our regex rules into CG rules with operators such as SELECT and REMOVE.

To give an example, in (1) we show the input to the tagger as a sequence of cohorts in CG-3 format. Each cohort consists of a surface form, shown within brackets inside quotes, followed by one or more readings. Each reading includes a lemma followed by one or more tags. Rules then apply to the input to remove impossible readings.

(1)
```
"<ཨ་ནེ་>"
        "ཨ་ནེ་" n.count
"<དང་>"
        "དང་" case.ass
        "དང་" cv.ass
        "དང་" v.invar
"<ཨ་ཁུ་>"
        "ཨ་ཁུ་" n.count
```

The word དང་ has three possible readings. In (1), དང་ is being used to coordinate two nouns, and so the correct reading is associative case or [case.ass]. Two separate rules remove [v.invar] and [cv.ass] as possible readings. A simplified version of the former rule is shown in (2).

(2)
```
REMOVE v.xxx (-1C n.xxx)
(0 ("<དང་>")) (1C n.xxx) ;
```

This rule removes [v.invar] from དང་ when it is sandwiched between two unambiguous (signified by C) nominals (n.xxx).

[1] Numbers in this section reflect a snapshot of the Classical Tibetan corpus as of 15 June, 2015.

The exercise of translating the rules into CG had no effect on the overall performance of the system, since it introduced no new rules or rule types. However, the exercise did put the rule tagger on a more secure footing for the future. Translated into CG, most of the rules can now be deciphered by linguists. Moreover, the general readability of CG means that linguists are now willing to take a stab at creating and modifying rules without the help of a technician.

2 Segmentation as syllable tagging

Tibetan orthography does not use whitespace or other means to mark the boundaries between words. However, the intersyllabic tsheg character (U+0F0B), resembling a dot, is used to mark the boundaries between orthographic syllables.[2] For example, consider the following sentence:

(3a) ངཡིས་མི་མང་པོ་བསད་
I killed many people.

(3b) ང་|p.pers
ཡིས་|case.agn
མི་|n.count
མང་པོ་|adj
བསད་|v.past

The correct segmentation for (3a) is shown in (3b), where each word appears on its own line, and the pipe character separates a word from its POS-tag. Only one word in this example consists of more than a single syllable.

Following Xue's (2003) general approach to Chinese word segmentation, Liu et al. (2011) propose that Tibetan word segmentation be recast as a syllable tagging problem. The task is then to tag each syllable according to its position in the word. We adopt their 6+2 tag set, which they say yields the best results given the average length of Tibetan words. We analyze (1) as:

(3c) ང་|S
ཡིས་|S
མི་|S
མང་|X
པོ་|E
བསད་|S

The S tag is given to syllables forming words on their own, while X and E mark the first and last syllables of disyllabic words. Longer words are marked with X-Y-E (trisyllabic), X-Y-Z-E (quadrisyllabic), and X-Y-Z-M*-E, with any number of M, for words of 5 or more syllables.

Two additional complex tags, SS and ES, are needed for the class of "abbreviated" syllables. These are situations of orthographic fusion where no intersyllabic tsheg separates the end of a word from the case marker or converb that follows it. Since such fusion only affects phonologically open syllables, whereas the same grammatical functions are indicated with different markers after closed syllables, we achieve consistency and avoid the unnecessary proliferation of POS-tags by treating such markers as their own words. (4) shows a form of the first-person pronoun with fused agentive case; because the syllable must be treated as two words, it is assigned the tag SS. Similarly in (5), meaning "of the many", because the genitive case marker must be pulled off from the word that precedes it to form its own word, the syllable is tagged ES instead of E.

(4) བས་ > ང|p.pers ས་|case.agn
བས་|SS

(5) མང་པོའི་ > མང་པོ་|adj འི་|case.gen
མང་|X
པོའི་|ES

It is important to remember that while the genitive marker shown in (5) only ever occurs in abbreviated syllables (and so only ever occurs in syllables tagged SS or ES), other abbreviated case markers and converbs look the same as natural word endings. For example, there are many possible analyses of the syllable མར་. It may continue an existing word (not shown), or be the beginning of a new word. The final ར་ may be the natural ending of the word, as in (6a), meaning "butter", or a case marker, as in (6b), meaning "down there".

(6a) མར་ > མར་|n.mass
མར་|S
(6b) མར་ > མ|d.dem ར་|case.term
མར་|SS

In summary, we use syllable tags to represent words as a sequence of tagged syllable tokens, as an alternative to joining syllables together into a sequence of word tokens. Special care must be taken when dealing with abbreviated syllables. As illustrated in (5), while such case markers and

[2] In this paper, henceforth, we refer to orthographic syllables with the term "syllable" . By doing so, we are not committing to analysing these units as syllables in the sense of phonological theory.

converbs are always counted as their own tokens in word-based tagging, they are fused with the preceding token in syllable-based tagging.

3 A syllable-based POS-tagger

In the next phase of the rule-based tagger, we modify the CG rules to operate over sequences of syllables rather than words. To do so, we change both the input and the rules themselves. Cohorts now become syllables instead of words. Syllable cohorts that belong to unambiguous words have only one reading, with one tag drawn from the 6+2 tagset and another drawn from the POS-tagset. Syllables belonging to ambiguous words will receive multiple readings, with each reading receiving the same 6+2 tag but a different POS-tag:[3]

(7)
```
"<ཨ་>"
        "ཨ་" X n.count
"<ནེ་>"
        "ནེ་" E n.count
"<དང་>"
        "དང་" S case.ass
        "དང་" S cv.ass
        "དང་" S v.invar
"<ཨ་>"
        "ཨ་" X n.count
"<ཀུ་>"
        "ཀུ་" E n.count
```

We recast rule (2) in syllabic terms, with only one difference from the original.

(8)
```
REMOVE v.xxx (-1C n.xxx)
(0 ("<དང་>") LINK T:IsWord)
(1C n.xxx) ;
```

Since the context word དང་ is monosyllabic, and since the POS-tag of a word is marked on all of its syllables, we can pretend that positions -1 and 1 are occupied by the words before and after དང་. In this and many other rules, we add a condition for monosyllabic wordhood.

(9)
```
TEMPLATE IsWord = 0C (S) ;
```

The template in (9) tests whether a syllable is an unambiguous monosyllabic word (and so tagged S) as opposed to being part of a word with the preceding or following syllable. Many rules, for example, target the syllable ས་, which is either negation [neg] or the noun "mother" [n.count]. Obviously, such rules should not apply to ས་ when it is part of a larger word such as བླ་ས་ "lama" [n.count].

Syllable-based rules become more interesting when they must manipulate multisyllabic words. For example, disyllabic nominals may be either verbal nouns or count nouns, but verbal nouns may not be followed by determiners. Simplifying somewhat by ignoring an exception to the rule, here is the original word-based rule:

(10)
```
REMOVE n.v.xxx (0 (n.count))
    (1C (d.plural)) ;
```

The syllable-based rule adds an extra condition when scanning for the determiner:

(11)
```
REMOVE n.v.xxx (0 (n.count))
    (T:NextInitial LINK 0C (d.plural)) ;
```

The template T:NextInitial ensures that the rule will correctly remove the impossible reading from both syllables of the nominal.

(12)
```
LIST Initial = S X SS ;
    TEMPLATE NextInitial=*1C Initial ;
```

Initials are defined as those syllables that can begin words. The template scans forward to the next syllable that can be an initial, and proceeds if that syllable is an initial on all of its readings. Since the syllables of a disyllabic nominal will be tagged X and E, the next initial for both is the first syllable of the word that follows.

A similar template, T:PrevFinal, scans left to grab the final syllable of the preceding word. Not unlike (11), (13) draws on the template in (14) to remove the tag [n.count] from both syllables of a verbal noun when it follows [case.term].

(13)
```
REMOVE (n.count) (T:PrevFinal
    LINK 0C (case.term)) (0 n.v.xxx) ;
```

(14)
```
LIST Final = S E SS ES ;
    TEMPLATE PrevFinal=*-1C Final ;
```

By linking conditions, we can scan more than one word in either direction. For example, the rule below selects the tag [d.det] for རང་ when it occurs in the context [n.count] [adj] རང་ མི་ འདུག་. Since Tibetan adjectives can be multisyllabic, it is necessary to seek past all syllables of the adjective to the final syllable of the preceding noun.

[3] This approach brings to mind Ng and Low's (2004) all-at-once character-based POS-tagger and segmenter for Chinese.

(15) SELECT (d.det) (-1C (adj) LINK T:PreviousFinal LINK 0C (n.count)) (0 ("<རང>") LINK T:IsWord) (1 ("<མི>") LINK T:IsWord) (2 ("<འདུག|?>"r) LINK T:IsWord) ;

As noted in the previous section, special care must be taken with abbreviated syllables. The genitive case marker, for instance, manifests as the abbreviated syllable འི, as in (5) above, or as one of several standalone syllables. Therefore, a rule such as the following which specifies ལས་ as a noun rather than ablative case if preceded by a genitive must accommodate both standalone (16) and abbreviated case (17).

(16) SELECT (n.count) (-1 ("<(གི|ཀྱི|གྱི)>"r) LINK T:IsWord) (0 ("<ལས>") LINK T:IsWord) ;

(17) SELECT (n.count) (T:PrevAbGen) (0 ("<ལས>") LINK T:IsWord) ;

(18) TEMPLATE IsAbGen = 0 ("<.+འི>"r) ; TEMPLATE PrevAbGen = T:PrevFinal LINK T:IsAbGen ;

In other cases, syllable-based tagging obviates the need for specific rules relating to abbreviated syllables. For example, (19) removes noun tags from ས་, provided that it is not preceded by end of sentence punctuation or by an intersyllabic tsheg. In addition to being a freestanding word meaning "earth", ས་ also marks agentive case when attached to open syllables, as in (4) above.

(19) REMOVE n.xxx (-1 tshegless) (0 (case.agn) LINK 0 ("<ས་?>"r)) ;

(20) SET shad = ("<[།༎]+>"r) ; SET tshegless = ("<.*[^་]>"r) - shad ;

No such rule is needed in the syllable-based rule tagger. If ས་ is attached to the preceding syllable, then that syllable will be tagged SS or ES, and the hypothesis that ས་ means "earth" will simply not arise.

4 Future directions

The syllable-based tagger adds complexity to the word-based tagger without improving its overall performance. So why use it?

In a traditional pipeline approach, tokenization or segmentation precedes part-of-speech tagging, with the output of the former process feeding the latter. This has the disadvantage that errors made at earlier stages in the pipeline propagate to later stages. The success of the pipeline is effectively limited by the quality of its initial component. So a Tibetan POS-tagger can only be as good as the word segmentation system that precedes it.

Another common limit of traditional pipelines is that different components often require that the data be represented in different ways. In practice, this can be an obstacle to component interaction. By bringing the data requirements of the word segmenter and the POS-tagger in line with each other, we hope to facilitate the development of more cooperative NLP components, including a joint approach to segmentation and tagging.[4]

References

Eckhard Bick and Tino Didriksen. 2015. CG-3 - Beyond classical constraint grammar. *Proceedings of the 20th Nordic Conference on Computational Linguistics (NODALIDA 2015)*, pages 31-39.

Edward Garrett, Nathan Hill and Abel Zadoks. 2014. A rule-based part-of-speech tagger for Classical Tibetan. *Himalayan Linguistics*, 13(1):9–57.

Hans van Halteren. 1999. Performance of taggers. In Hans van Halteren (ed.), *Syntactic Wordclass Tagging.* Springer: Netherlands, 81-94.

Huidan Liu, Minghua Nuo, Longlong Ma, Jian Wu, and Yeping He. 2011. Tibetan word segmentation as syllable tagging using conditional random field. *25th Pacific Asia Conference on Language, Information and Computation,* 168–177.

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? Word-based or character-based? *Proceedings of EMNLP.* Barcelona, Spain.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29-48.

[4] The Classical Tibetan corpus, as well as the various rule taggers described in this paper, are available on the following GitHub site: https://github.com/tibetan-nlp

Constraint Grammar as a SAT problem

Inari Listenmaa Koen Claessen
Chalmers University of Technology, Gothenburg, Sweden
{inari,koen}@chalmers.se

Abstract

We represent Constraint Grammar (CG) as a Boolean satisfiability (SAT) problem. Encoding CG in logic brings some new features to the grammars. The rules are interpreted in a more declarative way, which makes it possible to abstract away from details such as cautious context and ordering. A rule is allowed to affect its context words, which makes the number of the rules in a grammar potentially smaller. Ordering can be preserved or discarded; in the latter case, we solve eventual rule conflicts by finding a solution that discards the least number of rule applications. We test our implementation by parsing texts in the order of 10,000s–100,000s words, using grammars with hundreds of rules.

1 Introduction and previous research

Constraint Grammar (CG) (Karlsson et al.1995) is a relatively young formalism, born out of practical need for a robust and language-independent method for part-of-speech tagging. In this work, we present CG as a Boolean satisfiability (SAT) problem, and describe an implementation using a SAT solver. This is attractive for several reasons: formal logic is well-studied, and serves as an abstract language to reason about the properties of CG. Constraint rules encoded in logic capture richer dependencies between the tags than standard CG.

Applying logic to reductionist grammars has been explored earlier by (Lager1998; Lager and Nivre2001), but it was never adopted for use. Since those works, SAT solving techniques have improved significantly (Marques-Silva2010), and they are used in domains such as microprocessor design and computational biology—these problems easily match or exceed CG in complexity. Thanks to these advances, we were able to revisit the idea and develop it further.

Our work is primarily inspired by (Lager1998), which presents constraint rules as a disjunctive logic program, and (Lager and Nivre2001), which reconstructs four different formalisms in first-order logic. Other works combining logic to CG include (Eineborg and Lindberg1998) and (Sfrent2014), both using Inductive Logic Programming to learn CG rules from a tagged corpus.

2 CG as a SAT problem

Let us demonstrate our approach with the following example in Spanish.

```
"<la>"
        "el" det def f sg
        "lo" prn p3 f sg
"<casa>"
        "casa" n f sg
        "casar" v pri p3 sg
        "casar" v imp p2 sg
```

The ambiguous passage can be either a noun phrase, *la*<det> *casa*<n> ‘the house’ or a verb phrase *la*<prn> *casa*<v><pri><p3> ‘(he/she) marries her’. We add the following rules:

```
REMOVE prn IF (1 n) ;
REMOVE det IF (1 v) ;
```

Standard CG will apply one of the rules to the word *la*; either the one that comes first, or by some other heuristic. The other rule will not fire, because it would remove the last reading. If we use the cautious mode (`1C n` or `1C v`), which requires the word in the context to be fully disambiguated, neither of the rules will be applied. In any case, all readings of *casa* are left untouched by these rules.

The SAT solver performs a search, and starts building possible models that satisfy both constraints. In addition to the given constraints, we

have default rules to emulate the CG principles: an analysis is true if no rule affects it, and at least one analysis for each word is true—the notion of "last" is not applicable.

With these constraints, we get two solutions. The interaction of the rules regarding *la* disambiguates the part of speech of *casa* for free, and the order of the rules does not matter.

```
1) "<la>"
           "el" det def f sg
   "<casa>"
           "casa" n f sg

2) "<la>"
           "lo" prn p3 f sg
   "<casa>"
           "casar" v pri p3 sg
           "casar" v imp p2 sg
```

The most important differences between the traditional and the SAT-based approach are described in the following sections.

2.1 Rules disambiguate more

Considering our example phrase and rules, the standard CG implementation can only remove readings from the target word (`prn` or `det`). The SAT-based implementation interprets the rules as "determiner and verb together are illegal", and is free to take action that concerns also the word in the condition (`n` or `v`).

This behaviour is explained by simple properties of logical formulae. When the rules are applied to the text, they are translated into implications: `REMOVE prn IF (1 n)` becomes *casa*<n> $\Rightarrow$ $\neg$*la*<prn>, which reads "if the `n` reading for *casa* is true, then discard the `prn` reading for *la*". Any implication $a \Rightarrow b$ can be represented as a disjunction $\neg a \vee b$; intuitively, either the antecedent is false and the consequent can be anything, or the consequent is true and the antecedent can be anything. Due to this property, our rule translates into the disjunction $\neg$*casa*<n> $\vee$ $\neg$*la*<prn>, which is also equivalent to another implication, *la*<prn> $\Rightarrow$ $\neg$*casa*<n>. This means that the rules are logically flipped: `REMOVE prn IF (1 n)` translates into the same logical formula as `REMOVE n IF (-1 prn)`. A rule with more conditions corresponds to many rules, each condition taking its turn to be the target.

2.2 Cautious context is irrelevant

Traditional CG applies the rule set iteratively: some rules fire during the first iteration, either because their conditions do not require cautious context, or because some words are unambiguous to start with. This makes some more words unambiguous, and new rules can fire during the second iteration.

In SAT-CG, the notion of cautious context is irrelevant. Instead of removing readings immediately, each rule generates a number of implications, and the SAT solver tries to find a model that will satisfy them.

Let us continue with the earlier example. We can add a word to the input:

la casa grande 'the big house'

and a rule that removes verb reading, if the word is followed by an adjective:

```
REMOVE v IF (1 adj) ;
```

The new rule adds the implication *grande*<adj> $\Rightarrow$ $\neg$*casa*<v>, which will disambiguate *casa* to a noun[1]. As the status of *casa* is resolved, the SAT solver can now discard the model where *casa* is a verb and *la* is a pronoun and we get a unique solution with `det n adj`.

Contrast this with the behaviour of the standard CG. With the new rule, standard CG will also remove the verb reading from *casa*, but it is in no way connected to the choice for *la*. It all depends of the order of the two rules; if the `det` reading of *la* is removed first, then we are stuck with that choice. If we made the first rules cautious, that is, keeping the determiner open until *casa* is disambiguated, then we get the same result as with the SAT solver. Ideally, both ways of grammar writing should yield similar results; traditional CG rules are more imperative, and SAT-CG rules are more declarative.

2.3 Rules can be unordered

As hinted by the previous property, the SAT solver does not need a fixed order of the rules. Applying a rule to a sentence produces a number of clauses, and those clauses are fed into the SAT solver. However, in the unordered scheme, some

[1] Assuming that `adj` is the only reading for *grande*, it must be true, because of the restriction that at least one analysis for each word is true. Then the implication has a true antecedent (*grande*<adj>), thus its consequent ($\neg$*casa*<v>) will hold.

information is lost: the following rule sets would be treated identically, whereas in the traditional CG, only the first would be considered as a bad order.

```
1) SELECT v ;
   REMOVE v IF (-1 det) ;

2) REMOVE v IF (-1 det) ;
   SELECT v ;
```

Without order, both of these rule sets will conflict, if applied to an input that has sequence `det v`. The SAT solver is given clauses that tell to select a verb and remove a verb, and it cannot build a model that satisfies all of those clauses. To solve this problem, we create a variable for every instance of rule application, and request a solution where maximally many of these variables are true. If there is no conflict, then the maximal solution is one where all of these variables are true; that is, all rules take action.

In case of a conflict, the SAT solver makes it possible to discard only minimal amount of rule applications. Continuing with the example, it is not clear which instances would be discarded, but if the rules were part of a larger rule set, and in the context the REMOVE rule was the right one to choose, it is likely that the interaction between the desired rules would make a large set of clauses that fit together, and the SELECT rule would not fit in, hence it would be discarded.

This corresponds loosely to the common design pattern in CGs, where there is a number of rules with the same target, ordered such that more secure rules come first, with a catch-all rule with no condition as the last resort, to be applied if none of the previous has fired. The order-based heuristic in the traditional CG is replaced by a more holistic behaviour: if the rules conflict, discard the one that seems like an outlier.

We can also emulate order with SAT-CG. To do that, we enter clauses produced by each rule one by one, and assume the solver state reached so far is correct. If a new clause introduces a conflict with previous clauses, we discard it and move on to the next rule. By testing against gold standard, we see that this scheme works better with ready-made CGs, which are written with ordering in mind. It also runs slightly faster than the unordered version.

These three features influence the way rules are written. We predict that less rules are needed; whether this holds in the order of thousands of rules remains to be tested. On the one hand, getting rid of ordering and cautious context could ease the task of the grammar writer, since it removes the burden of estimating the best sequence of rules and whether to make them cautious. On the other hand, lack of order can make the rules less transparent, and might not scale up for larger grammars.

3 Evaluation

For evaluation, we measure the performance against the state-of-the-art CG parser VISL CG-3. SAT-CG fares slightly worse for accuracy, and significantly worse for execution time. The results are presented in more detail in the following sections.

3.1 Performance against VISL CG-3

We took a manually tagged corpus[2] containing approximately 22,000 words of Spanish news text, and a small constraint grammar[3], produced independently of the authors. We kept only SELECT and REMOVE rules, which left us 261 rules. With this setup, we produced an ambiguous version of the tagged corpus, and ran both SAT-CG and VISL CG-3 on it. Treating the original corpus as the gold standard, the disambiguation by VISL CG-3 achieves F-score of 82.6 %, ordered SAT-CG 81.5 % and unordered SAT-CG 79.2 %. We did not test with other languages or text genres due to the lack of available gold standard.

We also tested whether SAT-CG outperforms traditional CG with a small rule set. With our best performing and most concise grammar[4] of only 19 rules, both SAT-CG and VISL CG-3 achieve a F-score of around 85 %. This experiment is very small and might be explained by overfitting or mere chance, but it seems to indicate that rules that work well with SAT-CG are also good for traditional CG.

[2] https://svn.code.sf.net/p/apertium/svn/branches/apertium-swpost/apertium-en-es/es-tagger-data/es.tagged

[3] https://svn.code.sf.net/p/apertium/svn/languages/apertium-spa/apertium-spa.spa.rlx

[4] https://github.com/inariksit/cgsat/blob/master/data/spa_smallset.rlx

# rules	SAT-CG$_u$	SAT-CG$_o$	VISL CG-3
19	39.7s	22.1s	4.2s
99	1m34.1s	1m14.9s	6.1s
261	2m54.1s	2m31.6s	10.7s

Table 1: Execution times for 384,155 words.

3.2 Execution time

The worst-case complexity of SAT is exponential, whereas the standard implementations of CG are polynomial, but with advances in SAT solving techniques, the performance in the average case in practice is more feasible than in the previous works done in 90s–00s. We used the open-source SAT solver MiniSat (Eén and Sörensson2004).

We tested the performance by parsing Don Quijote (384,155 words) with the same Spanish grammars as in the previous experiment. Table 1 shows execution times compared to VISL CG-3; SAT-CG$_u$ is the unordered scheme and SAT-CG$_o$ is the ordered. From the SAT solving side, maximisation is the most costly operation. Emulating order is slightly faster, likely because the maximisation problems are smaller. In any case, SAT does not seem to be the bottleneck: with 261 rules, the maximisation function was called 147,253 times, and with 19 rules, 132,255 times, but the differences in the execution times are much larger, which suggests that there are other reasons for the worse performance. This is to be expected, as SAT-CG is currently just a naive proof-of-concept implementation with no optimisations.

4 Applications and future work

Instead of trying to compete with the state of the art, we plan to use SAT-CG for grammar analysis[5]. There has been work on automatic tuning of hand-written CGs (Bick2013), but to our knowledge no tools to search for inconsistencies or suboptimal design.

The sequential application of traditional CG rules is good for performance and transparency. When a rule takes action, the analyses are removed from the sentence, and the next rules get the modified sentence as input. As a downside, there is no way to know which part comes directly from the raw input and which part from applying previous rules.

A conflict in an ordered scheme can be defined as a set of two or more rules, such that applying the first makes the next rules impossible to apply, regardless of the input. We can reuse the example from Section 2.3:

```
SELECT v ;
REMOVE v IF (-1 det) ;
```

The first rule selects the verb reading everywhere and removes all other readings, leaving no chance for the second rule to take action. If the rules are introduced in a different order, there is no conflict: the REMOVE rule would not remove verb readings from all possible verb analyses, so there is a possibility for the SELECT rule to fire.

Ordered SAT-CG can be used to detect these conflicts without any modifications, as a side effect of its design. After applying each rule, it stores the clauses produced by the rule and commits to them. In case of a conflict, the program detects the particular rule that violates the previous clauses, with the sentence where it is applied. Thus we get feedback which rule fails, and on which particular word(s).

Unordered SAT-CG with maximisation-based conflict solving is not suitable for this task: the whole definition of conflict depends on ordering, and the unordered scheme deliberately loses this information. On a more speculative note, an unordered formalism such as Finite-State Intersection Grammar (Koskenniemi1990) might benefit from the maximisation-based technique in conflict handling.

Finally, we intend to test for conflicts without using a corpus. Let us illustrate the idea with the same two rules, `SELECT v` and `REMOVE v IF (-1 det)` in both orders. Assume we have the tag set {`det, n, v`}, and we want to find if there exists an input such that both rules, applied in the given order, remove something from the input. There are no inputs that satisfy the requirement with the first order, but several that work with the second, such as the following:

```
"<w1>"
        det
        v
"<w2>"
        n
        v
```

Thus we can say that the first rule order is conflicting, but the second one is not. Implementing and testing this on a larger scale is left for future work.

[5]We thank Eckhard Bick for the idea.

5 Conclusions

SAT-solvers are nowadays powerful enough to be used for dealing with Constraint Grammar. A logic-based approach to CG has possible advantages over more traditional approaches; a SAT solver may disambiguate more words, and may do so more precisely, capturing more dependencies between tags. We experimented with both ordered and unordered rules, and found the ordered scheme to work better with previously written grammars. For future direction, we intend to concentrate on grammar analysis, especially finding conflicts in constraint grammars.

Acknowledgments

We thank Eckhard Bick, Tino Didriksen, Francis Tyers and Anssi Yli-Jyrä for comments and suggestions, as well as the anonymous reviewers and everyone who participated in the discussion at the CG workshop.

References

Eckhard Bick. 2013. ML-Tuned Constraint Grammars. In *Proceedings of the 27th Pacific Asia Conference on Language, Information and Computation*.

Niklas Eén and Niklas Sörensson. 2004. An Extensible SAT-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing*, volume 2919 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

Martin Eineborg and Nikolaj Lindberg. 1998. Induction of constraint grammar-rules using progol. In David Page, editor, *Inductive Logic Programming*, volume 1446 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg.

Fred Karlsson, Atro Voutilainen, Juha Heikkilä, and Arto Anttila. 1995. *Constraint Grammar: a language-independent system for parsing unrestricted text*, volume 4. Walter de Gruyter.

Kimmo Koskenniemi. 1990. Finite-state parsing and disambiguation. In *Proceedings of the 13th Conference on Computational Linguistics - Volume 2*, COLING '90. Association for Computational Linguistics.

Torbjörn Lager and Joakim Nivre. 2001. Part of speech tagging from a logical point of view. In *Logical Aspects of Computational Linguistics, 4th International Conference, LACL 2001, Le Croisic, France, June 27-29, 2001, Proceedings*.

Torbjörn Lager. 1998. Logic for part of speech tagging and shallow parsing. In *Proceedings of the 11th Nordic Conference on Computational Linguistics*.

João Marques-Silva. 2010. Boolean Satisfiability Solving: Past, Present & Future. Presentation given at the Microsoft Research International Workshop on Tractability, Cambridge, UK, July 5–6.

Andrei Sfrent. 2014. Machine Learning of Rules for Part of Speech Tagging. Master's thesis, Imperial College London, United Kingdom.

Using weighted finite state morphology with VISL CG-3—Some experiments with free open source Finnish resources

Tommi A Pirinen
Ollscoil Chathair Bhaile Átha Cliath
CNGL—School of Computing
Dublin City University, Dublin 9
tommi.pirinen@computing.dcu.ie

Abstract

Traditionally, the coupling of finite state morphology and constraint grammar has been strictly rule-based, making binary distinctions between allowed and disallowed readings, however, in the recent years much of the research in the finite state morphologies has adapted the contemporary paradigm of statistically weighted analysis. This is reflected in current versions of free and open source morphology of Finnish, omorfi, in the finite state morphology part. In this paper we examine two strategies of making use of the weights as a part of VISL CG-3 pipeline. We evaluate the results intrinsically on small sample of analyses we have disambiguated by hand ourselves, and extrinsically on the effect it has on the rule-based machine translation of that text using the freely available open source translator, apertium-fin-eng.

1 Introduction

In the recent years, use of statistical information in computational linguistics has gained much interest, with systems like hunpos (Halácsy et al., 2007), moses (Koehn et al., 2007) etc. being the main points of interest of most research in the field. In finite state morphology as well as constraint grammars, extensions to handle probabilities are recent and scarcely documented (Lindén and Pirinen, 2009; Bick, 2009). In this paper we experiment with an existing weighted finite state morphology of Finnish (Pirinen, 2011)[1] with VISL CG-3. For CG we have adapted Fred Karlsson's Finnish CG rules to omorfi's tag set, however, the rules were written for completely different analyser, which results in relatively low quality and high level of ambiguity at the current level. We estimate that salvaging these rules for the current version of analysis would require a substantial re-writing effort. In the meanwhile, there are a lot of easy targets that correctly trained statistical analyser can already deal with without extra effort. E.g., one large difference we assume between the analyser these CG rules were written for and omorfi's are that omorfi contains a huge number of proper nouns, dialectal and sub-standard forms, and rare language, animal etc. names, that are left ambiguous. It is obvious for a human reader that these words are very unlikely and given most corpora we expect them to be highly penalised as well.

The main goal of this experiment is to create a functional pipeline out of weighted finite-state analysis and current version of the constraint grammar. There are obvious conflicts between the statistically driven ranked hypotheses approach and the strictly deleting approach of the current constraint grammar, which may limit usefulness of our current method of combining these two information sources.

The rest of the paper is structured as follows: In section 2 we explain our starting point and current pipelines for morphological analysis, disambiguation and machine translation. In section 3 we explain various approaches we tried to include and combine weight data from the weighted finite-state analysers into VISL CG-3 and finally into machine translation. In section 4 we describe our experiments and how we measured the workability of our approach. In section 5 we show the results of the experiment. In section 6 we perform error analy-

[1] https://github.com/flammie/omorfi/

sis, compare our work with other existing approaches and lay out the future work. Finally in section 7 we summarise the conclusion of the experiments.

2 Background

Our starting point for this experiment is such that we had a modern, weighted finite-state morphology (Beesley and Karttunen, 2003; **?**) implementation of Finnish morphology in omorfi (**?**). This morphology has rudimentary support for probabilistic weighting of surface forms or analyses using corpora-based unigram training approach. However, with the lack of high quality free and open source corpora compatible with omorfi analyses means that it is distributed with very basic linguist-written weights on the analysis side. For the main purpose of this experimentation we deemed this sufficient, to get the weights working through the pipeline at all.

On the other hand we had a free and open source, mature and large CG grammar by Fred Karlsson, that needed conversion to omorfi compatible tagging format, as well as some changes from CG-1 syntax to VISL CG-3.[2]

The fact that the CG rules from Karlsson were built using very different analyser than ours also played a large role in our decision to combine the weighted approach to with pure constraint grammar approach: the rule-writers of the original grammar had not seen large portion of the ambiguities introduced by larger, more varied lexicon of omorfi, including things like dialects, large inventories of proper nouns and unlikely but attested readings like plural cases of singular personal pronouns. For example, in the story we use for reference in our translation experiments, the sentence initial common words like "Mutta" (but) and "Koira" (dog) are also proper nouns, but also proper nouns like "Mari" have been added a common noun reading (slang for marihuana). Obviously these are not dealt with in the original ruleset as they have not appeared as ambiguities to the writers of th rules.

[2]even though CG-1 and VISL CG-3 are possibly are mostly compatible, we found that some things may have started working better when changing to more conventional VISL CG-3 constructions

3 Methods

To first convert the original CG-1 ruleset to omorfi format analyses, we went through the rules by hand from beginning to end. This resulted in a ruleset where only a subset of rules matched to any constructs in the analysed texts. To further improve the quality and fix a lot of conversion errors we made use of the new VISL CG-3 features `no-inline-sets`. With help of this feature we got most of the ambiguous word-forms at least to match some of the rules, which hopefully means conversion has not too many tag formatting mismatch errors at the very least. The resulting ruleset with weight-based rule integrated is available from omorfi git repository.

To feed omorfi analyses into VISL CG-3 we have extended the python interface of omorfi to output CG stream format analyses, with omor style `[FEATURE=VALUE]` tags mapped into more conventional CG style tags, mostly of form `VALUE`. There are number of deviations to this of course, most notable being the `WEIGHT=` feature, which is turned into VISL CG-3 *numeric tag*. Other special conversions include things like usage, dialect and such lexical information, which are all included in angle bracket tags following VISL CG-3 conventions. Omorfi python interface also performs some case mangling (uppercasing, lowercasing, title-casing and removing title case) that seems to be similar as CG-1 rules seem to expect to appear in some angle-bracketed tags, so we have tried to map these to the readings in the original ruleset, with limited success.

The probabilities in omorfi are provided by the underlying HFST (Lindén et al., 2011) system as a floating point number based on the finite-state implementation of a tropical semiring. This weight can be based on negative logarithms of probabilities of the word-forms, lemmas, analyses etc., as well as linguist-defined arbitrary values. For the purposes of this experiment we only used the linguist-defined values that are neatly in range of 0.1—32. This simplifies the scaling of the weights introduced by VISL CG-3 processing as we only have to scale against known range instead of e.g. combinations of negative logarithms' maxima. As noted earlier in section 2, we use the default setting which is based on

linguist-approximated tag likelihoods. Since VISL CG-3 does not support floating point numbers, e.g. 0.1, we output weight in a numeric tag multiplied by a 100 before rounding them down and turning into a tag of the form `<W=weight>`, where *weight* is the multiplied weight. This is sufficient for the coarse weights that default analyser produces, and in line what e.g. `cg-conv` does when it treats stream formats containing decimal data to be converted into numeric tags.

The basic support for numeric tag processign in VISL CG-3 is done by the `SELECT (<W=MIN>)` statement. If applied as a sole rule to result of omorfi to VISL CG-3 conversion it exactly like traditional weighted finite state morphology producing 1-best analysis. When combined into existing ruleset, we add this into a last, separate `SECTION`, in order to integrate some weight handling to CG iterations.

One long-term goal of this experimentation was to use VISL CG-3 also as a part of morphological analysis pipeline that produces n-best lists in same manner as weighted finite-state analyser does. To make this work, we take the output of VISL CG-3's cg-proc in trace mode before converting it back to an n-best list. There are multiple possible strategies to use readigns for deleted analyses as weights again. With this experiment, we have simply gone with adding the line number of the rule, this reflects the fact that later rules in the file are more risky and less ambiguous. Ideally however, we would like to annotate the rules using rule name labels, such as "usually", "dangerous" to denote e.g. multipliers for such rules. Furthermore, it is likely that it is not exactly the line number, but rather the section number, that is relevant for the rule likelihood, due to way linguists and rulewriters will organise rules within sections into blocks of related rules where ordering within and between blocks may not be important.

4 Experimental Setup

For analysis we use the python API to omorfi version 20150326, to turn the analyses into the format understood by VISL CG 3. We use a version Fred Karlsson's Finnish CG found in apertium's repository.[3], with the tag set manually converted to match omorfi's,[4] however, given the amount of ambiguous names of tags and sets and lists in the grammar, there may be some conversion errors left. The system is tested with VISL CG-3 version 0.9.9.10730, compiled from Gentoo packaging.[5]

To test the functionality of our combination of weighted finite-state analyser and VISL CG-3, we analyse a short text that we have manually disambiguated and measure the quality of analyses. The source of the text is found in the apertium's SVN repository.[6] For the purpose of this experiment, we have manually tokenised the text before processing it with omorfi. In addition to analysis we use the results of analyses in apertium's Finnish-English machine translator, and measure the translation quality. This way we can ensure that the gold annotation has not been selected to best fit our results but is actually the semantically most fitting one. The gold annotations can also be found in the omorfi git repository.

To perform evaluations we used simple python script that performs string comparisons of the gold file lines between the lines starting with `"<` ignoring empty lines and the ADDed CLB tags. The machine translation analysis was performed against current apertium-fin-eng ruleset and the reference translation in their svn, with standard machine translation metrics as measured by NIST's `mteval-13a.pl`, which is the standard BLEU metric of machine translation (**?**).

5 Evaluation

We first evaluated the analysers against the gold standard in table 1. We use simple metrics of *Recall* and *Precision*, defined as $\textbf{Recall} = \frac{\text{Correct}}{\text{Gold}}$, where Correct is number of correct readings and Gold is number of gold readings, and $\textbf{Precision} = \frac{\text{Correct}}{\text{All}}$, where All is number of all readings given by the disam-

[3] `http://sourceforge.net/p/apertium/svn/HEAD/tree/nursery/apertium-fin-eng/apertium-fin-eng.fin-eng.rlx`

[4] `https://github.com/flammie/omorfi/tree/master/src/vislcg3`

[5] `https://github.com/flammie/flammie-overlay/tree/master/sci-misc/vislcg3`

[6] `http://sourceforge.net/p/apertium/svn/HEAD/tree/nursery/apertium-fin-eng/texts/tarina.fin.text`

Rules	*Precision*	*Recall*
Weights	60	99
Rules	78	91
Combination	80	90

Table 1: Precision and recall of different combinations of weighted morphology and rules.

Rules	*Precision*	*Recall*
Weights	60	99
Rules	78	91
Combination	80	90

Table 2: Precision and recall of different combinations of weighted morphology and rules.

biguation scheme. The row *Weights* stands for CG with only select weighted best applied, the row *Rules* stands for only converted CG ruleset applied, and row *Combination* uses both.

The resulting analyses is then converted to format expected by apertium for machine translation and evaluated for machine translation quality in table 2.

6 Discussion

First of all, we note that the quality differences with adding weights has diminished from the version prior to conference and current version. This is largely due to newer version released in the workshop containing features that greatly improved the tag matching of the converted ruleset. Following this result we can say that the weights are most useful when the rules are not as high coverage, i.e. early stages of development or, as in this case, conversion process.

Nevertheless, the overall effect of combining weights has still improvements to exactly the shortcomings noted in the introduction as problems of the mismatching morphologies. In error evaluation, the cases that are affected by rules are mostly in derivation and productive compounding, but also some marginal cases that are not covered by rules.

For future work we are aiming to use the n-best list version of the result in a real-world application pipeline.

7 Conclusion

We have implemented a VISL CG-3 output on top of existing weighted finite-state analysis of Finnish language and tested that it works combined with VISL CG-3. We have successfully included this combination as a part of apertium machine translation pipeline. We note that weighted finite-state analysis can be easily combined with VISL CG 3 and results in an increased accuracy.

Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement PIAP-GA-2012-324414 (Abu-MaTran).

References

Kenneth R Beesley and Lauri Karttunen. 2003. Finite-state morphology: Xerox tools and techniques. *CSLI, Stanford*.

Eckhard Bick. 2009. Introducing probabilistic information in constraint grammar parsing. In *Proceedings of Corpus Linguistics 2009*.

Péter Halácsy, András Kornai, and Csaba Oravecz. 2007. Hunpos: an open source trigram tagger. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 209–212. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180. Association for Computational Linguistics.

Krister Lindén and Tommi Pirinen. 2009. Weighting finite-state morphological analyzers using HFST tools. In Bruce Watson, Derrick Courie, Loek Cleophas, and Pierre Rautenbach, editors, *FSMNLP 2009*, July.

Krister Lindén, Erik Axelson, Sam Hardwick, Tommi A Pirinen, and Miikka Silfverberg. 2011. Hfst—framework for compiling and applying morphologies. *Systems and Frameworks for Computational Morphology*, pages 67–85.

Tommi A Pirinen. 2011. Modularisation of Finnish finite-state language description—towards wide collaboration in open source development of morphological analyser. In *Proceedings of Nodalida*, volume 18 of *NEALT proceedings*.

Anaphora resolution experiment with CG rules

Tiina Puolakainen
University of Tartu, Estonia
tiina.puolakainen@ut.ee

Abstract

This article describes an ongoing work - an experiment on anaphora resolution in Estonian newspaper texts using Constraint Grammar (CG) rules. The personal, demonstrative and relative pronouns are chosen for resolution at the first stage of the research. As morphological information and syntactic relations play an important role in identifying anaphoric relations, syntactically analyzed text is used as an input for the CG rules.

1 Introduction

The experiment is performed on a subtask of coreference resolution task, considering only pronouns, in particular personal, demonstrative and relative pronouns, some of the latter can serve as interrogative pronouns as well in particular sentences. Reference is exophoric if the referred entity is brought into the textual space and pronoun refers to those objects of the real world that the speakers can see or imagine. Reference is endophoric if the referred item has been mentioned earlier in the text or there is an intention to do so immediately. Endophoric reference can be called anaphoric if the antecedent of the pronoun has occurred previously in the text or cataphoric if it will follow the pronoun (Pajusalu, 1996).

The antecedent of a pronoun in at text is usually a noun or a noun phrase or a noun in an adpositional phrase, but it can be also an adjective in case of adjectival pronoun or it may refer to an entire clause in case pronoun refers to a situation expressed by the whole clause. Finally the antecedent of a pronoun can be also another pronoun - then it can refer further to it's antecedent and so compose a chain of pronouns hopefully ending by a meaningful reference.

The latest version of the CG formalism (Karlsson et al., 1995) - VISL CG3 framework (Bick, 2000) is chosen for the needs of the experiment because this framework is very flexible and precise at the same time. It enables to write as general rules as possible and as specific as needed - in that sense it can be considered a programming language that is proved to be suitable and successful for many language processing tasks.

Preliminary work consisted of extracting of synonyms and synosets with hyponym relation for some task-specific notions as speech-verbs and animate-inanimate nouns from the Estonian Wordnet (Vider and Orav, 2005). The extracted sets however could not be used entirely because they included much more submeanings of the desired concepts than could be affordable. The resulting sets were filtered manually and then included into CG grammar as a word lists used by the CG rules.

2 Anaphora resolution

The procedure takes a syntactically analyzed Estonian text from the Estonian Dependency Treebank (Muischnek et al., 2014). Sentences have been analyzed by syntactic parser (Muischnek, Müürisep and Puolakainen, 2014) and then manually checked and corrected by a linguist. It is possible to use just automatically analyzed sentences but in this case more errors occur due to remaining morphological and syntactical ambiguities and also errors made by the syntactical analyzer.

First the pronouns are identified in the text by adding a special tag to the word analysis. Then

the relation identification rules are applied for each pronoun.

CG grammar for identification of coreferences consist of three types of rules according to the type of pronoun. Personal pronouns are also differentiated by the person (1st, 2nd, 3rd) and all pronouns by the number (singular, plural). Demonstrative pronouns referring to more general situation expressed by a clause as well as relative-interrogative pronouns in interrogative sentences receive a reference relation to a predicate of the corresponding clause. Special attention has to be paid to the direct speech and as a special case - a dialog or interview-style text. The additional rules have to be added to enable to deal with different styles of presenting such text, currently only the style that is encountered in a training corpus is maintained by the rules.

Taking advantage of known syntactical functions, usually subject and object are preferred by the rules if choosing the relation candidate as subject and object as core arguments are more likely to carry the main theme of the story. But in specific situations other considerations are used by the rules as for example the cases of complements of speech-verbs indicating the theme of the talk while searching for antecedent of demonstrative pronoun.

An animate-inanimate sets were worth of using them but should be further examined and subcategorized for fitting their aim more precisely. The difficulty in using these sets arises due to the quite free grammatical choice of corresponding pronoun, especially in a case of inanimate concept that can be referred with personal pronoun. The most frequent example for that is a concept of some organisation, that can be referred both as inanimate and animate substance, in the latter meaning the people of that organisation. In this case even number 'agreement' between pronoun and its antecedent may not hold. Pronoun can be in plural and it's antecedent in singular also for other generalizations. According to Pajusalu (1996), prototypically *tema* 'he/she' refers to animate and *see* 'it' to inanimate entity, but the real use of pronoun system is much more complicated. For example, quite common is to use *tema* to refer to more concrete entity and *see* for more abstract one. But it depends very strongly on particular topic, for example music can be considered concrete when talking about particular composition and an abstract in general, it depends also on the formality of the situation. From two relatively similar entities in the animate-inanimate and abstract-concrete axes first is usually referred as *tema* and second as *see*. There are definitely many more nuances of different usage of pronoun system and all kinds of metaphors also cause this kind of difficulties for recognizing coreferences.

3 Evaluation

For evaluating the rules a newspaper text of 2080 words was taken from Estonian Dependency Treebank. The test benchmark consisted of four stories from different newspapers and was not especially selected to contain very complicated coreference relations. 150 CG rules for anaphora resolution were applied to the syntactically analyzed text and then manually checked for correctness of marked reference. The text contained 101 pronouns, of which 79 received correct reference relation that means a 78% recall. Applying the same procedure to the same but automatically analyzed text increases the amount of errors from 22 to 28 giving recall of 72%. Table 1 shows results of the evaluation according to the type of the pronouns, numbers in parentheses indicate the corresponding values then applied on automatically analyzed text. Most difficulties are imposed by demonstrative pronouns as they have also the widest spectrum of entities they can refer to, including exophoric reference where any suggestion of an antecedent in the text become wrong. The easiest is resolution of reference of relative pronouns, if only they are correctly distinguished from the interrogative usage.

	Demonstrative	Personal	Relative	Total
Correct no	20 (18)	30 (27)	29 (28)	79 (73)
Correct %	59 (53)	81 (73)	97 (93)	78 (72)
Incorrect no	14 (16)	7 (10)	1 (2)	22 (28)
Incorrect %	41 (47)	19 (27)	3 (7)	22 (28)
Total	34	37	30	101

Table 1. Evaluation results of anaphora resolution.

An example of a sentence (1) with successfully recognized reference relations both in a case of a treebank sentence and an automatically syntactically analyzed sentence:

(1) *Digitaalühenduse (1) saavad need (→3) kliendid (2), kes (→2) seda (→1) soovivad (3).*

Digital-connection-sg.gen (1) receive that-pl.nom (→3) clients (2) who (→2) this-sg.prt (→1) want (3).

'The digital connection (1) can be received by whose (→3) clients (2), who (→2) want (→3) that (→1).'

'That' points to 'digital connection', 'who' to 'clients' and 'whose' refer to the subordinate clause 'who want that' specifying the kind of 'whose' clients.

In the following sentence (2) two relations were recognized correctly and third remained unrelated with both types of input as rules could not determine the exact referred situation in the previous context:

(2) *Ma (→2 sentences ahead) (1) ei tea, kui hästi see (→none) mul (→1) õnnestub.*

I (→2 sentences ahead) (1) not know how well that (→none) I-sg.ade (→1) succeeds.

'I (→2 sentences ahead) (1) do not know, how well it (→none) (for me (→1)) succeeds.'

Situations with multiple suitable candidates and/or no sufficiently good candidate are difficult to handle. As the rules are applied individually, one by one, each of them has to be quite careful for their decision. In some circumstances no rule in the set of rules formulated so far can decide the correct antecedent. Mostly this indicates 'gaps' that can be filled by formulating new rules.

(3) *Puutüved (1) on miljoneid aastaid (1') vastu pidanud tänu sellele (→2), et neid (→1/1') ümbritses (2) kiht liiva (3), mis (→3) võis olla tekkinud mõnest tugevast liivatormist.*

Tree-trunks (1) are million-pl.prt year-pl.prt (1') against hold-ppp thanks this-sg.all (→2) that this-pl.prt (→1/1') enclosed (2) coat sand-sg.prt (3) that (→3) might have evolved some-sg.ela strong-sg.ela sandstorm-sg.ela.

'The treetrunks (1) have survived for millions of years (1') due to the fact (→2) that they (→1/1') have been enclosed (2) by a coat of sand (3) that (→3) might have evolved as a result of a strong sandstorm.'

In sentence (3) all three relations are correctly recognized using a treebank sentence as a input, but makes one mistake using automatically syntactically analyzed sentence: 'they' is found to refer to 'millions of years' instead of 'The trunks' due to remained morphological ambiguity in the sentence.

One particular source of errors in automatically syntactically analyzed text is remaining ambiguity between *tema* 'he/she' and *see* 'it' pronouns that have some homonymous forms or in the worse case the error made during disambiguation of these forms. There is no good solution so far for this kind of errors. On the one hand disambiguation module needs the information of referenced entity to make right decision choosing between two pronouns, but on the other hand coreference resolution module expects that the right decision is already done by disambiguation module. Also wrong case and subsequently incorrectly chosen syntactical function can cause additional errors in automatic mode.

One more source of errors is the distance between pronoun and it's antecedent. Choosing far distances could allow to find also far relations but at the same time it would introduce more errors there unrelated entities would be marked as related. The CG framework allows to choose a window size for the working distance of the rules and after some experiments a window of 7 sentences was chosen as an optimal working load. This caused one error in test set where the correct antecedent was in the 9th sentence ahead.

4 Using synonym relations

The hypothesis for this experiment was that antecedent of a demonstrative pronoun can be situated in a very similar context, namely containing synonym words. In order to check this hypothesis all synonyms in the text at a distance of maximally 7 sentences from each other were found using Estonian Wordnet hierarchy of synosets and up to 3 layers of hypo- and hyperonym relations. After that rules were applied if encountered pronoun and potential antecedent in the context of synonyms.

An example (4) of a successful application of such rule: *hõõgus* 'smouldered' is found to be a synonym of *põleks* 'could burn' and due to that the subject pronoun *see* 'it' in one sentence was related with it's antecedent - subject in the sentence with synonym predicate.

(4) ... *turvas* (SUBJ) hõõgus (SYN) eredalt isegi päevavalguses ...

... *peat* (SUBJ) smouldered (SYN) brightly even in day light ...

... (some sentences)

Vabalt põleks (SYN) *see* (SUBJ) veel vähemalt 4 nädalat.

It (SUBJ) could burn (SYN)at least 4 weeks further.

In one more successful example (5) pronoun *neist* 'form them' was correctly related to an antecedent NP, but, in fact, verbs that are found to be synonyms in these sentences are not used in their synonym submeanings – *käima* as 'go' and *tulema* in it's modal sense 'have to', not it's main sense 'to come'.

(5) Jooksu *intensiivsus ja maht* käivad (SYN) käsikäes.

The *intensity and amount* of running go (SYN) hand in hand.

Üht *neist* suurendades tuleb (SYN) teist vähendada.

Increasing one *of them* (you) have to (SYN) reduce the other.

Overall, this experiment didn't bring an improvement, but did even worse, because found synonym and close hypo- and hyperonym relations appeared to be mostly too loose or general and didn't account for different submeanings and caused too many errors.

5 Conclusions

The results of the small experiment on usual newspaper text are good for the beginning - in 72-78% of cases pronouns receive correct reference relation. The experiment shows the main difficulties and limitations of approach encountered during anaphora resolution and gives the directions for further work, including the research of wider range of coreference types besides pronouns. The main aim is to use other sources of semantic relations and methods of distributive semantics besides wordnet that can help to solve the task. The many-to-many synonym relations of wordnet synosets cannot be efficiently realized by the means of CG framework and need to be handled in a separate preprocessing step.

References

Eckhard Bick. 2000. *The Parsing System "Palavras": Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework.* Aarhus University Press, Aarhus, UK.

Fred Karlsson, Atro Voutilainen, Juha Heikkilä, Arto Anttila. 1995. *Constraint Grammar: A Language-Independent System for Parsing Running Text.* Natural Language Processing, No 4. Mouton de Gruyter, Berlin and New York.

Kadri Muischnek, Kaili Müürisep and Tiina Puolakainen. 2014. Dependency Parsing of Estonian: Statistical and Rule-based Approaches. In *Proceedings of the Sixth International Conference: Human Language Technologies – The Baltic Perspective.* Amsterdam: IOS Press, (Frontiers in Artifical Intelligence and Appications; 268), pages 111-118.

Kadri Muischnek, Kaili Müürisep, Tiina Puolakainen, Eleri Aedmaa, Riin Kirt, Dage Särg. 2014. Estonian Dependency Treebank and its annotation scheme. In *Proceedings of 13th Workshop on Treebanks and Linguistic Theories (TLT13).* Tübingen, pages 285-291.

Renate Pajusalu. 1996. Pronoun Systems of Common Estonian and Estonian Dialects in Contrastive Perspective. *Estonian Typological Studies I.* Tartu, pages 145-164.

Kadri Vider and Heili Orav. 2005. Estonian wordnet and Lexicography. *Symposium on Lexicography XI. Proceedings of the Eleventh International Symposium on Lexicography.* Lexicographica, Series Maior 115. Max Niemeyer Verlag, Tübingen, pages 549-555.

Abbreviations used in glosses

ade adessive case
all allative case
gen genitive case
ela elative case
nom nominative case
pl plural
prt partitive case
ppp past participle
sg singular

A preliminary constraint grammar for Russian

Francis M. Tyers
HSL-fakultehta,
UiT Norgga árktalaš universitehta,
N-9018 Romsa
francis.tyers@uit.no

Robert Reynolds
HSL-fakultehta,
UiT Norgga árktalaš universitehta,
N-9018 Romsa
robert.reynolds@uit.no

Abstract

This paper presents preliminary work on a constraint grammar based disambiguator for Russian. Russian is a Slavic language with a high degree of both in-category and out-category homonymy in the inflectional system. The pipeline consists of a finite-state morphological analyser and constraint grammar. The constraint grammar is tuned to be high recall (over 0.99) at the expense of low precision.

1 Introduction

This paper presents a preliminary constraint grammar for Russian. The main objective of the constraint grammar is to produce a high recall grammar to serve as input into other natural language processing tasks. There are two reasons to maintain high recall. First, one of the primary applications for this constraint grammar is computer-assisted language learning. In the domain, erroneous analyses can lead to significant frustration for learners. Therefore, it is important to limit disambiguation to cases that can be resolved with high confidence. Second, it is frequently the case that competing readings can be distinguished only by considering idiosyncratic collocational information. For such cases, we expect that probabilistic approaches are both more effective and simpler to implement.

The paper is laid out as follows: section 2 presents a review of the literature on Russian language processing; section 3 gives an overview of ambiguity in Russian; section 4 describes our analysis pipeline; section 5 gives an account of our development process; section 6 presents an evaluation of the system, and sections 7 and 8 present future work and conclusions.

2 Review of literature

State-of-the-art morphological analysis in Russian is primarily based on finite-state technology (Nozhov, 2003; Segalovich, 2003).[1] Almost without exception, all large-scale morphological transducers of Russian are based on the forward-looking *Grammatical Dictionary of Russian* (Zaliznjak, 1977). This dictionary gives fine-grained morphological specifications for more than 100 000 words, including inflectional endings, morphophonemic alternations, stress patterns, exceptions, and idiosyncratic collocations. We developed a morphological transducer based on Zaliznjak's dictionary.[2] This finite-state transducer (FST) generates all possible morphosyntactic readings of each wordform, regardless of its frequency or probability. Because Russian is a relatively highly inflected language, broad coverage is important, but widespread homonymy leads to the generation of many spurious readings, as discussed in Section 3 below. Because of this, one of the foundational steps in Russian natural language processing is homograph disambiguation.

3 Ambiguity in Russian

We identify three different types of morphosyntactic ambiguity: intraparadigmatic, morphosyntactically incongruent, and morphosyntactically congruent. The following examples make use of word stress ambiguity to illustrate each kind of ambiguity.[3] *Intraparadigmatic* ambiguity refers to homo-

[1] Machine-learning approaches have also been successfully applied to Russian, most notably by Sharoff et al. (2008).

[2] Our transducer is implemented using a two-level morphology (Koskenniemi, 1984), and can be compiled using either `xfst` (Beesley and Karttunen, 2003) or `hfst` (Linden et al., 2011)

[3] Written standard Russian does not typically indicate stress position, but knowing stress position is essential for pronunciation. A recent study by Reynolds and Tyers (2015) found that about 7.5% of morphosyntactic ambiguity in a cor-

graphic wordforms belonging to the same lexeme, as shown in (1).

(1) Intraparadigmatic homographs
   a. те́ла *téla* 'body.$_{\text{SG-GEN}}$'
   b. теля́ *telá* 'body.$_{\text{PL-NOM}}$'

The remaining two types of ambiguity occur between lexemes. *Morphosyntactically incongruent* ambiguity occurs between homographs that belong to separate lexemes, and whose morphosyntactic values are different, as shown in (2).

(2) Morphosyntactically incongruent homographs
   a. на́шей *nášej* 'our.$_{\text{F-SG-GEN/DAT/LOC...}}$'
      нашей́ *našéj* 'sew on.$_{\text{IMP-2SG}}$'
   b. доро́га *doróga* 'road.$_{\text{N-F-SG-NOM}}$'
      дорога́ *dorogá* 'dear.$_{\text{ADJ-F-SG-PRED}}$'

Morphosyntactically congruent ambiguity occurs between homographs that belong to separate lexemes, and whose morphosyntactic values are identical, as shown in (3).

(3) Morphosyntactically congruent homographs
   a. за́мок *zámok* 'castle.$_{\text{SG-NOM}}$'
      замо́к *zamók* 'lock.$_{\text{SG-NOM}}$'
   b. за́мков *zámkov* 'castle.$_{\text{PL-GEN}}$'
      замко́в *zamkóv* 'lock.$_{\text{PL-GEN}}$'
      etc.

Table 1 shows the prevalence of each kind of ambiguity. The first column shows the proportion of all tokens in a corpus that have each kind of ambiguity. The second column shows what proportion of ambiguous tokens exhibit each kind of ambiguity. Note that these proportions do not sum to 100%, since a given token may exhibit more than one kind of ambiguity. For example, the wordform *zamkov* has the readings given in (4).

(4) a. замок^1+N+Msc+Inan+Pl+Gen
   b. замок^2+N+Msc+Inan+Pl+Gen
   c. замковый+A+Msc+Sg+Pred

The ambiguity between (4-a) and (4-b) is morphosyntactically congruent, and the ambiguity between (4-a)/(4-b) and (4-c) is morphosyntactically incongruent, so this wordform would be counted for both categories in Table 1.

pus of Russian resulted in stress position ambiguity.

Table 1 shows that most morphosyntactic ambiguity in unrestricted Russian text is rooted in intraparadigmatic and morphosyntactically incongruent ambiguity. Detailed part-of-speech tagging with morphosyntactic analysis can help disambiguate these forms. On the other hand, morphosyntactically congruent ambiguity represents only a very small percentage of ambiguous wordforms, and instead of detailed part-of-speech tagging, it can be resolved by means of word sense disambiguation. Because of this difference, we leave morphosyntactically congruent ambiguity to future work.

Type	**all tokens**	**ambig. tokens**
Intraparadigm.	59.0%	90.9%
Incongruent	27.7%	42.7%
Congruent	1.2%	1.8%

Table 1: Frequency of different types of morphosyntactic ambiguity in unrestricted text

4 Pipeline

4.1 Morphological analyser

The morphological transducer used in this study is primarily based on Zaliznjak's *Grammatical dictionary of Russian*, including the 2001 version's appendix of proper nouns. It also includes neologisms from Grishina and Lyashevskaya's *Grammatical dictionary of new Russian words*, which is intended to be a supplement to Zaliznjak's dictionary with words found in the Russian National Corpus.[4] Example (5) gives some examples of the FST's output.

(5) a. новый<adj><m><nn><sg><nom>
      'new'
   b. автомат<n><m><nn><sg><nom>
      'automaton, sub-machine gun'

4.2 Disambiguation rules

The constraint grammar is composed of 299 rules which are divided into four categories: Safe, Safe heuristic, Heuristic, and Syntax labeling. The distribution of rules is shown in Table 2.

The philosophy is that Safe rules should represent real constraints in the language. Examples might be that a preposition cannot directly precede a finite verb or that prepositional case requires a preceding preposition.

[4] http://dict.ruslang.ru/gram.php

	SELECT	REMOVE	MAP
Safe	16	34	–
Safe heuristic	89	76	–
Heuristic	26	52	–
Syntax labelling	–	–	6

Table 2: The 299 rules in the grammar are separated into four sections depending on rule reliability.

Safe heuristic rules should deal with highly frequent tendencies in the language. For example remove a genitive at the beginning of a sentence if it is capitalised and there is no verb governing the genitive found to the right and there is also no negated verb to the right. This rule relies on the fact that if the genitive is in first position in the sentence it cannot modify anything before, and no preposition can be governing it. This kind of rule often relies on completeness of sets, in this case the set of verbs that can take a genitive complement.

Heuristic rules are those which we do not consider linguistic constraints, but express preferences, often dealing with overgeneration or overspecification in the morphological transducer. For example, remove the verbal adverb reading of такая, which could be the feminine singular nominative of такой 'such' or the verbal adverb of такать 'say *well*...'.

Given a large hand-annotated corpus we believe that most of the heuristic rules would be better replaced with information learnt from the corpus through stochastic methods.

5 Development process

A common approach taken when writing constraint grammar rules is to apply the existing rule set to a new text, write new rules to deal with the ambiguities, then apply the rules to a hand-annotated corpus to see how often the rule disambiguated correctly (Voutilainen, 2004).

Due to the lack of a hand-annotated corpus compatible with our morphological analyser, we adopted a slightly modified technique. We picked a random text from the Russian Wikipedia,[5] ran it through the morphological analyser, wrote rules, and then ran the rules on the whole Wikipedia corpus. For each rule, we collected around 100 example applications and checked them. If a rule selected the appropriate reading in all cases, we included it in the *safe* rule set, if it removed an appropriate reading in less then three cases, then we included it in the *safe heuristic* rule set. Otherwise we either discarded the rule or included it in the heuristic rule set.

[5]The Russian Wikipedia was chosen as a testing corpus as it is the largest, freely licensed corpus of Russian available on the internet. It is not representative of Russian texts as a whole.

6 Evaluation

6.1 Corpus

In order to evaluate the grammar we hand-annotated 10,150 words of Russian text from Wikipedia articles, public domain literature and freely-available news sources. The annotated texts are available online under the CC-BY-SA licence.[6]

Hand-annotation proceeded as follows: The text was first morphologically analysed, and then an annotator read through the output of the morphological analyser, commenting out the readings which were not appropriate in context. This annotated text was then checked by a second annotator.

We chose to annotate our own texts as opposed to using a well-known hand-annotated corpus such as the Russian National Corpus (RNC) for two main reasons: the first was that the RNC is not freely available; the second was that the standards for tokenisation, part-of-speech and morphological description are different from our morphological analyser.

Table 3 gives a quantitative evaluation of the performance of our CG on the test corpus.

6.2 Qualitative evaluation

In this section, we give a qualitative evaluation of errors made by the CG.

Bad linguistics: In some cases a rule did not take into account grammatical possibilities in the language. e.g. Two simple rules such as

- `REMOVE Det IF (0 Det OR Pron) (1C Ne) ;`
- `REMOVE Det IF (0 Det OR Pron) (1 Cm LINK 1 CC OR CS) ;`

did not take into account the possibility of having a postposed determiner as in

- ...а может быть и раньше, и факт этот не раз поражал меня ...

[6]https://svn.code.sf.net/p/apertium/svn/languages/apertium-rus/texts/

```
"<В>"
    "в" pr

"<ноябре>"
    "ноябрь" n m nn sg prp

"<1994>"
    "1994" num

"<года>"
    "год" n m nn sg gen SELECT:r462
;   "год" n m nn pl nom fac SELECT:r462

"<в>"
    "в" pr

"<Танзании>"
    "Танзания" np al f nn pl acc
    "Танзания" np al f nn sg prp
;   "Танзания" np al f nn pl nom REMOVE:r424
;   "Танзания" np al f nn sg dat REMOVE:r433
;   "Танзания" np al f nn sg gen REMOVE:r433

"<начал>"
    "начало" n nt nn pl gen
    "начать" vblex perf tv past m sg
;   "начать" vblex perf iv past m sg REMOVE:r769

"<работу>"
    "работа" n f nn sg acc

"<Международный>"
    "международный" adj m an sg nom
    "международный" adj m nn sg acc

"<трибунал>"
    "трибунал" n m nn sg acc
    "трибунал" n m nn sg nom

"<по>"
    "по" pr

"<Руанде>"
    "Руанда" np al f nn sg prp
    "Руанда" np al f nn sg dat
"<.>"
    "." sent
```

Figure 1: Example output from the morphological analyser and constraint grammar for the sentence В ноябре 1994 года в Танзании начал работу Международный трибунал по Руанде. "The work of the International Tribunal for Rwanda started in Tanzania in November 1994." The input ambiguity is 1.76 readings per word and the output ambiguity is 1.38 readings per word. Recall is 1.0 and precision is 0.72. Figure 2 shows the rules that fired for this example sentence.

```
### Safe

SELECT:r462 Gen IF (0 Year) (-1 Num LINK -1 Months LINK -1 Pr/V);
  # Select genitive reading of 'года' if there is a numeral immediately
  # to the left, before that there is a month and before that there is
  # the preposition 'в'.

REMOVE:r424 Nom IF (-1C Pr) ;
  # Remove nominative case if there is a word which can only be a
  # preposition immediately to the left.

REMOVE:r433 NGDAIP - Acc - Prp - Loc IF
                (-1C* Pr/V OR Pr/Na
                  BARRIER (*) - Adv - Comp - DetIndecl - ModAcc - ModPrp);
  # Remove all cases apart from accusative, preposition and locative
  # if 'в' or 'на' are found to the left and are unambiguous. The barrier
  # is anything that cannot be found inside a noun phrase.

### Safe heuristic

REMOVE:r769 IV IF (0 TV OR IV) (1C Acc) (NOT 1 AccAdv);
  # Remove an intransitive reading of a verb if the next word can only
  # be accusative and is not in the set of nouns which can be used
  # adverbially in accusative.
```

Figure 2: Some example rules from the grammar.

Domain	Tokens	Precision	Recall	F-score	Ambig. solved
Wikipedia	7,857	0.506	0.996	0.671	44.92%
Literature	1,652	0.473	0.984	0.638	42.95%
News	642	0.471	0.990	0.638	41.60%
Average	10,150	0.498	0.994	0.663	44.39%

Table 3: Results for the test corpora.

- . . . and maybe even earlier, and fact **this** not once surprised me . . .

or a interposed parenthetical as in

- Но какие, однако же, два разные создания, точно обе с двух разных планет!
- But what, **exactly** , two different creatures, just both from two different planets!

Bad rule: In some cases a rule was simply incorrectly specified. For example, the following rule was designed to solve the ambiguity between short-form neuter adjectives and adverbs

- `REMOVE A + Short IF (-1C Fin OR Adv OR A) (0C Short OR Adv) ;`

However there is no reason why we should prefer an adverb over an adjective after an adverb,

- . . . потому что совсем неприятно проснуться в гробу под землею.
- . . . because [it is] really **unpleasant** to wake up in a coffin under the ground.

Incomplete barrier: Some rules suffered from incomplete barriers, which is something that would benefit from a more systematic treatment.

- `REMOVE NGDAIP - Acc - Prp - Loc IF (-1C* Pr/V OR Pr/Na BARRIER (*) - Adv - Comp - DetIndecl - ModAcc - ModPrp) ;`

here the rule removes the nominative reading of the adjective to leave the accusative reading because the preposition в 'in' is found preceeding.

- В 1960-х электрифицированные высокоскоростные железные дороги появились в Японии и некоторых других странах.
- In the 1960's **electrified** high-speed railways appeared in Japan and some other countries.

Incomplete set: In some cases the rule was a good generalisation, but made use of a set which was incomplete. For example:

- `REMOVE Dat IF (NOT 0 Prn/Sebe) (NOT 0 Anim OR Cog OR Ant) (NOT 0 Pron) (NOT 1* V/Dat) (NOT -1* V/Dat) (NOT -1* Prep/Dat) (NOT -1C A + Dat) ;`

the set `V/Dat` does not contain the verb противопоставлять 'opposed to' which takes a dative argument.

- В связи с этим ортодоксальности стали противопоставлять ересь.
- In connection with this **orthodoxy** was opposed to heresy.

Rule interaction: The strong accusative rule below causes incorrect behaviour in the rule to remove transitivity readings

- `REMOVE TV - Pass IF (NOT 1* Acc) (NOT -1* Acc) ;`
- `REMOVE Acc IF (-1C Fin + IV) (NOT 0 AccAdv) ;`

Consider the following example where может 'can' is tagged as intransitive, the second rule fires removing the accusative reading of его 'him', and thus given the lack of accusative reading, найти 'find' is disambiguated as intransitive instead of transitive.

- Она смотрит везде, но не может его найти.
- She looks around, but she cannot **find him**.

Difficult linguistics: Dealing with participles with arguments is challenging in the case that the arguments of the participle share the same government as the main verb.

- `REMOVE IV IF (0 TV OR IV) (1C Acc) (NOT 1 AccAdv) ;`

Here Ваню и Машу 'Vanja and Maša' are the object of видит 'sees' and not играющих 'playing', although both verbs can take accusative object.

- Их мама внутри дома с кошкой, она смотрит в окно и видит играющих Ваню и Машу.

- Their mother is inside the house with the cat, she looks through the window and sees Vanja and Maša **playing**.

This kind of error would ideally be resolved with semantic knowledge.

6.3 Task-based evaluation

The constraint grammar described in this paper has been applied to the task of automatic word stress placement (Reynolds and Tyers, 2015). This task is especially relevant for Russian language learners, because vowels are pronounced differently depending on their position relative to stress position. For example, the word *molokó* 'milk' is pronounced /məlʌkɔ/, where each instance of the letter *o* corresponds to a different vowel sound. Russian has complicated patterns of shifting stress, which are difficult for learners to master. Almost 99% of wordforms with ambiguous stress position can be disambiguated morphosyntactically, so a constraint grammar can potentially resolve most stress ambiguity indirectly. The results of Reynolds and Tyers (2015) show that our constraint grammar overcomes about 42% of the ambiguity relevant to stress ambiguity in unrestricted text.

6.4 Combining with a statistical tagger

Given that just over half of all ambiguity remains after running our preliminary constraint grammar and that for many applications unambiguous output is necessary, we decided to experiment with combining the constraint grammar with a statistical tagger to resolve remaining ambiguity. Similar approaches have been taken by previous researchers with Basque (Ezeiza et al., 1998), Czech (Hajič et al., 2001; Hajič et al., 2007), Norwegian (Johannessen et al., 2011; Johannessen et al., 2012), Spanish (Hulden and Francom, 2012), and Turkish (Oflazer and Tür, 1996).

We follow the voting method described by Hulden and Francom (2012). We used the freely available `hunpos` part-of-speech tagger (Halácsy et al., 2007). We performed 10-fold cross validation using our evaluation corpus, taking 10% for testing and 90% for training, and experimented with three configurations:

- `HMM`: the `hunpos` part-of-speech tagger with its default options
- `HMM+Morph`: as with `HMM` but incorporating the output of our morphological analyser (see section 4.1) as a full form lexicon.
- `HMM+Morph+CG`: we submitted the output from `HMM+Morph` and the constraint grammar to a voting procedure, whereby if the constraint grammar left one valid reading, we chose that, otherwise if the constraint grammar left a word with more than one reading, we chose the result from the `HMM+Morph` tagger.

As can be seen from Figure 3, incorporating the constraint grammar improves the performance of the HMM tagger, an improvement of nearly 5% in accuracy, similar to that reported by Hulden and Francom (2012) for the same amount of training data. In Figure 3, it appears that the HMM alone is much more dependent on training corpus size than the voting setup, which improves very little between a training corpus size of 5,000 and 9,000.

Our constraint grammar also has a much lower precision as a result of the ambiguity remaining in the output. Similarly, the final accuracy is below the state of the art for Russian. For instance, Sharoff et al. (2008) report a maximum accuracy of 95.28% using the TnT tagger. Note, however, that this model was trained on a much larger corpus – over five million tokens – which is not freely available.

7 Future work

We have a number of plans for future work, the first of which is increasing the precision of the grammar without decreasing recall. Secondly we would like to add syntactic function labelling and dependency parsing. For the dependency parser we plan to reuse the Giellatekno dependency grammar as in (Antonsen et al., 2010).

The development workflow could also be improved, for example during the testing of each rule we could save the correct decisions of the grammar. This would give us a partially-disambiguated development corpus, which could be gradually used to build up a gold-standard corpus, and which could also be used for regression testing to ensure that new rules added do not invalidate the correct decisions of previously written rules.

Also it is worth noting that although Russian has a great deal of non-free resources, this paper also presents a method which is promising

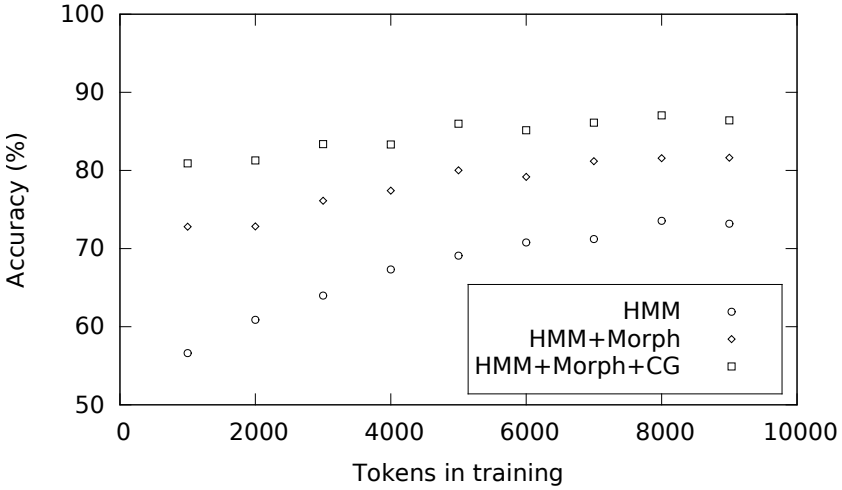

Figure 3: Learning curve for three taggers, `hunpos` with no lexicon, `hunpos` with a lexicon, and `hunpos` with a lexicon and the Russian constraint grammar in a voting set up.

for smaller or lesser-resourced Slavic languages such as Sorbian, Rusyn or Belarusian. Instead of hand-annotating a large quantity of text, it may be more efficient to work on grammatical resources — such as a morphological analyser and constraint grammar — and use them alongside a smaller quantity of high-quality annotated text.

8 Conclusions

This paper has presented a preliminary constraint grammar for Russian, where rules have been assigned to sections based on observations of performance on a non-gold corpus. The constraint grammar is high recall (over 0.99) and improves the performance over a trigram HMM-based tagger. It also shows state-of-the-art performance for the stress-placement task.

Acknowledgments

We are grateful to Koen Claessen for insightful discussion, as well as three anonymous reviewers who gave thoughtful feedback on an earlier version of this paper. All remaining errors are our own.

References

Lene Antonsen, Linda Wiechetek, and Trond Trosterud. 2010. Reusing grammatical resources for new languages. In *Proceedings of the International conference on Language Resources and Evaluation LREC2010*, pages 2782–2789.

Kenneth R Beesley and Lauri Karttunen. 2003. *Finite-state morphology: Xerox tools and techniques*. CLSI, Stanford.

Nerea Ezeiza, Iñaki Alegria, José María Arriola, Rubén Urizar, and Itziar Aduriz. 1998. Combining stochastic and rule-based methods for disambiguation in agglutinative languages. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 380–384. Association for Computational Linguistics.

Jan Hajič, Pavel Krbec, Pavel Květoň, Karel Oliva, and Vladimír Petkevič. 2001. Serial combination of rules and statistics: A case study in czech tagging. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 268–275. Association for Computational Linguistics.

Jan Hajič, Jan Votrubec, Pavel Krbec, Pavel Květoň, et al. 2007. The best of two worlds: Cooperation of statistical and rule-based taggers for czech. In *Proceedings of the Workshop on Balto-Slavonic Natural Language Processing: Information Extraction and Enabling Technologies*, pages 67–74. Association for Computational Linguistics.

Péter Halácsy, András Kornai, and Csaba Oravecz. 2007. Hunpos: An open-source trigram tagger. In *Proceedings of the 45th annual meeting of the ACL*, pages 209–212.

Mans Hulden and Jerid Francom. 2012. Boosting statistical tagger accuracy with simple rule-based grammars. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation*.

Janne Bondi Johannessen, Kristin Hagen, André Lynum, and Anders Nøklestad. 2011. OBT+Stat: Evaluation of a combined CG and statistical tagger. In Eckhard Bick, Kristin Hagen, Kaili Müürisep,

and Trond Trosterud, editors, *Proceedings of the NODALIDA 2011 Workshop Constraint Grammar Applications*, volume 14, pages 26–34, Riga, Latvia. NEALT.

Janne Bondi Johannessen, Kristin Hagen, André Lynum, and Anders Nøklestad. 2012. Obt+stat: A combined rule-based and statistical tagger. In Gisle Andersen, editor, *Exploring Newspaper Language: Using the Web to Create and Investigate a Large Corpus of Modern Norwegian*, pages 51–66. John Benjamins Publishing.

Kimmo Koskenniemi. 1984. A general computational model for word-form recognition and production. In *Proceedings of the 10th International Conference on Computational Linguistics*, COLING '84, pages 178–181, Stroudsburg, PA, USA. Association for Computational Linguistics.

Krister Linden, 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*, volume Vol. 100 of *Communications in Computer and Information Science*, pages 67–85. Springer.

Igor Nozhov. 2003. Морфологическая и синтаксическая обработка текста (модели и программы) *[Morphological and Syntactic Text Processing (models and programs)] also published as* Реализация автоматической синтаксической сегментации русского предложения *[Realization of automatic syntactic segmentation of the Russian sentence]*. Ph.D. thesis, Russian State University for the Humanities, Moscow.

Kemal Oflazer and Gökhan Tür. 1996. Combining hand-crafted rules and unsupervised learning in constraint-based morphological disambiguation. In *Proceedings of the ACLSIGDAT Conference on Empirical Methods in Natural Language Processing*, pages 69–81, Philadelphia, PA, USA.

Robert Reynolds and Francis Tyers. 2015. Automatic word stress annotation of Russian unrestricted text. In *Main conference proceedings from NODALIDA 2015*, Vilnius, Lithuania. NEALT.

Ilya Segalovich. 2003. A fast morphological algorithm with unknown word guessing induced by a dictionary for a web search engine. In *International Conference on Machine Learning; Models, Technologies and Applications*, pages 273–280.

Serge Sharoff, Mikhail Kopotev, Tomaž Erjavec, Anna Feldman, and Dagmar Divjak. 2008. Designing and evaluating a Russian tagset. In *Proceedings of the Sixth Language Resources and Evaluation Conference, LREC 2008*, Marrakech.

Atro Voutilainen. 2004. Hand crafted rules. In H. van Halteren, editor, *Syntactic Wordclass Tagging*, pages 217–246. Kluwer Academic.

Andrej Anatoljevič Zaliznjak. 1977. Грамматический словарь русского языка: словоизменение: около 100 000 слов *[Grammatical dictionary of the Russian language: Inflection: approx 100 000 words]*. Изд-во "Русский язык".