NEALT PROCEEDINGS SERIES
VOL. 14

Proceedings of the NODALIDA 2011 workshop

Constraint Grammar Applications

May 11, 2011
Riga, Latvia

Editors
Eckhard Bick, Kristin Hagen, Kaili Müürisep, Trond Trosterud

NORTHERN EUROPEAN ASSOCIATION FOR LANGUAGE TECHNOLOGY

Proceedings of the NODALIDA 2011 workshop in
Constraint Grammar Applications

NEALT Proceedings Series, Vol. 14

ISSN 1736-6305

Published by
Northern European Association for Language
Technology (NEALT)
http://omilia.uio.no/nealt

Electronically published at
Tartu University Library (Estonia)
http://dspace.utlib.ee/dspace/handle/10062/19231

Volume Editors
Eckhard Bick, Kristin Hagen, Kaili Müürisep, Trond Trosterud

Contents

Constraint Grammar Applications

Eckhard Bick
University of Southern Denmark

This paper collection presents the contributions to the 2011 NoDaLiDa Constraint Grammar workshop.

The NoDaLiDa CG workshop has now seen 3 editions, and become the main physical forum for the exchange of ideas and results in the field of Constraint Grammar at the research level. As organizers we were pleased to note a growing number of participants and a consistently high quality of contributions. The last years have seen extensions of CG usage along several lines, both to the formalism as such, and in terms of areas of application. Thus, the CG3 formal language now allows the emulation of very diverse grammatical approaches, covering besides the traditional topological and dependency frameworks also the use of probabilistic, generative and unification techniques. Most strikingly, however, is the potential CG has shown in the applicational arena, where it is now successfully used across a wide range of languagetechnological issues, such as machine translation, grammar checking, dialogue systems and lexicography. Finally, CG continues to be used for the production of linguistic research and teaching resources, such as annotated corpora in general, and treebanks in particular.

The papers in this collection provide an insider's view on some of these developments. Annotation topics cover both treebanks (Voutilainen) and spoken data (Donnelly & Deuchar), and MT features prominently in the applicational area, touching both on methodology such as valency portability (Arriola & Wiechetek) and grand-scale projects such as Wikipedia translation (Bick). Within the field of CG theory, hybrid solutions (Johannessen et al.) and compiler optimization through FST methods (Peltonen) are explored. Parsing efficiency is also the central theme in Yli-Jyrä's paper presenting a full CG compiler implementation with an inward deterministic method.

Finally, it should be noted that CG's rule-based approach continues to allow researchers to tackle also smaller languages, where the quality or indeed feasibility of machine-learning suffers from the lack of existing training resources. Thus, Basque (Arriola & Wiechetek), Sami (Antonsen & Trosterud) and Esperanto (Bick) are represented in this collection of workshop papers.

On behalf of the organizing team,
Eckhard Bick

The papers

Lene Antonsen and Trond Trosterud: *Next to nothing – a cheap South Saami disambiguator*

Eckhard Bick: WikiTrans: *The English Wikipedia in Esperanto*

Kevin Donnelly and Margaret Deuchar: *Using constraint grammar in the Bangor Autoglosser to disambiguate multilingual spoken text*

Janne Bondi Johannessen, Kristin Hagen, Andrè Lynum, Anders Nøklestad: *OBT+Stat: Evaluation of a combined CG and statistical tagger*

Janne Peltonen: *A Finite State Constraint Grammar Parser*

Atro Voutilainen: *FinnTreeBank: Creating a research resource and service for language researchers with Constraint Grammar*

Anssi Yli-Jyrä: *An Efficient Constraint Grammar Parser based on Inward Deterministic Automata*

Linda Wiechetek and Jose Mari Arriola: *An Experiment of Use and Reuse of Verb Valency in Morphosyntactic Disambiguation and Machine Translation for Euskara and North Sámi*

PROGRAMME COMMITTEE

- Eckhard Bick, Syddansk universitet
- Kristin Hagen, Universitetet i Oslo
- Kaili Müürisep, Tartu Ülikool
- Trond Trosterud, Universitetet i Tromsø

WORKSHOP PROGRAMME

Wednesday, May 11, Riga

09.00 - 09.10
Trond Trosterud: *Opening statements*

09.10 - 09.40
Jose Mari Arriola and Linda Wiechetek: *An experiment of Use and Reuse of Verb Valency in Morphosyntactic Disambiguation and Machine Translation for Basque and North Sámi*

09.40 - 10.10
Eckhard Bick: *The English Wikipedia in Esperanto*

10.10 - 10.30
Coffee break

10.30 - 11.00
Lene Antonsen and Trond Trosterud: *Next to nothing – a cheap South Sami disambiguator*

11.00 - 11.30
Kevin Donnelly and Margaret Deuchar: *Using constraint grammar in the Bangor Autoglosser to disambiguate multilingual spoken text*

11.30 - 12.00
Jackson Ssekiryango: *Towards a Luganda Constraint Grammar*

12.00 - 13.00
Lunch break

13.00 - 13.30
Anssi Yli-Jyrä: *An Efficient Constraint Grammar Parser based on Inward Deterministic Automata*

13.30 - 14.00
Atro Voutilainen, Tanja Purtonen, Kristiina Muhonen and Mikaela Kumlander: *FinnTreeBank: Creating a research resource and service for language researchers with Constraint Grammar*

14.00 - 14.30
Janne Peltonen: *Finite state constraint grammar parser*

14.30 - 15.00
Janne Bondi Johannessen, Kristin Hagen, Andrè Lynum and Anders Nøklestad: *OBT+Stat: Evaluation of a combined CG and statistical tagger*

15.00 - 15.30
Coffee break

15.30 - 16.30
Tino Didriksen: *Latest news*

Next to nothing – a cheap South Saami disambiguator

Lene Antonsen
University of Tromsø
Norway
lene.antonsen
@uit.no

Trond Trosterud
University of Tromsø
Norway
trond.trosterud
@uit.no

Abstract

The goal of this article was to show that even a small constraint grammar may achieve results good enough to be used as a lemmatiser. The result shows that a rule set of 115 CG rules is efficient enough to give a lemmatisation accuracy (lemma + POS identification) of 1.056 for open POS.

1 Introduction

Lemmatising is important for a whole range of language technology applications. Morphology-rich languages get better word alignment, and both dictionary and terminology work need lemmatisation in order to be able to search for words in texts in reliable ways.

Constraint grammars are widely recognised for achieving deep syntactic analyses very close to the gold standard, but at the expense of requiring carefully crafted rule sets of several thousand rules (Karlsson et. al, 1995). The goal of this article is to investigate whether a small rule set may achieve a more restricted task, namely POS and lemma disambiguation.

1.1 Lemmatising

Deciding whether two word forms belong to the same lemma or not might be problematic. In order to do that, we first define the parts of speech of the language by morphosyntactic means. Which lexeme a given word form belongs to, will then follow from the overall POS structure. For us, lemmatising thus means finding the correct lexeme for each word form. Our research shows that even a small constraint grammar may achieve results good enough to be used as a lemmatiser.

Homonymy in the Uralic languages is more often than not confined to paradigm-internal homonymy. Two homonym word forms usually express different grammatical words of the same lexeme, and not homonym word forms of different lexemes. This means that even a partial disambiguation may be helpful for lemmatising, even though it fails in resolving all the grammatical ambiguities.

2 Relevant features of South Saami grammar

South Saami is, like the other Saami languages, a Uralic language. Typologically, it has a medium-size morphology, with 8 cases, 2 numbers for nouns, and 9 person-number values, 2 moods and 2 tenses for verbs, in addition to several infinite verbforms and a productive derivational morphology. The relatively agglutinative morphology is combined with a rather complex morphophonology (Sammallahti, 1998).

The most important morphophonological process is an Umlaut system consisting of 7 different vowel series and 6 different morphophonologically defined contexts. Other processes include diphthong simplification processes and suffix alternations depending upon the underlying foot structure.

Compared to the other Saami languages, South Saami has relatively little morphological ambiguity. On average, each reading receives 1.6 analyses, as compared to 2.6 analyses for North Saami.

3 Derivations

In the Saami languages there is much derivation, for all the open word classes. In our transducer lexicon (at http://giellatekno.uit.no), many of the derivations are lexicalized. Since more work has been done for North Saami than for the other languages, there are more lexicalisations in the North Saami lexicon than in the Lule and South Saami ones. In the output from the morphological analyser, there are dynamic analyses, in addition to the possibly lexicalized

one, as shown in Figure 1.

bájkálattjat (Lule Saami) ('locally')
`bájkke N Der1 Der/lasj A Der2 Der/at Adv`

báikkálaččat (North Saami) ('locally')
`báiki N Der1 Der/laš A Der2 Der/at Adv`
`báikkálaš A Der2 Der/at Adv`
`báikkálaččat Adv`

Figure 1: The morphological analysis of derived words may differ for the *sme* and *smj* analysers.

When extracting term pairs from parallel corpora, the challenge is to extract the lemmas in one language against the non-lexicalised lemma + derivation affix series in the other.

The algorithm is as follows:

1. Choose the lexicalized reading if there is one
2. If there is no lexicalised reading, choose the derived one with the fewest number of derivational affixes.

The Lule Saami word *bájkálattjat* means 'locally', and is derived from the adjective meaning 'local' which is derived from the noun meaning 'place'. In this case, word alignment between Lule Saami and North Saami gives the following alignment: *bájkke* 'place' = *báikkálaččat* 'locally'.

A better solution is to glue the derivation tags to the lemma, so the word alignment process will align *bájkke N Der1 Der/lasj A Der2 Der/at Adv* to *báikkálaččat Adv*. Figure **??**, Matt. 9.8., gives an example of lemmatised text with derivation tags.

original text:
`Muhto olbmot ballagohte go oidne dán, ja sii máidno Ipmila gii lea addán olbmuide dakkár fámu.`

lemmatised text:
`muhto olmmoš ballat+V+Tv+Der3+Der/goahti go oaidnit dát , ja son máidnut Ipmil gii leat addit olmmoš dakkár fápmu .`

'But people began to be afraid when they saw it, and they prised God which had given the people such a power.'

Figure 2: The lemmatised text contains derivation tags.

4 South Saami as part of a larger Saami analyser

The Saami languages have different morphological and morphophonological processes, and therefore separate morphological transducers are built for each language.

The output of the morphological analysers is then disambiguated in separate modules for each language. Due to different homonymy patterns of the languages, different rules apply. North Saami needs many rules in order to resolve the homonymy between accusative and genitive case. In Lule Saami, this type of homonymy is restricted to the personal pronouns, and in South Saami it does not exist at all.

The mapping of syntactic tags to conjunctions, subjunctions and finite and non-finite verbs is done at an early stage in the North and Lule Saami disambiguation files because these tags are used for sentence boundary detection, which is crucial for disambiguation of e.g. case forms.

However, the mapping of most of the syntactic tags is done in a common module shared by all three Saami languages, as shown in Table 1. The annotation is based on 49 syntactic tags.[1]. Due to the relatively free word order in Saami, a fairly large number of tags is needed.

The rules in the syntactic analyser refer to morphological tags and sets of lemmas (e.g. the `TIME` set contains lemmas that denote time adverbials), which are language specific. The disambiguator adds language tags (<sme>, <smj>, <sma> for North, Lule and South Saami, respectively) to all morphological analyses. When a lemma is identified as belonging to a certain language, language-specific rules and language-specific exceptions are triggered. E.g., in South Saami, the copula is often omitted in existential and habitive sentences, which means there is no finite verb in the sentence. In North Saami, a sentence without a finite verb is analysed as a fragment or an elliptic sentence, which is not appropriate for South Saami. Furthermore, the habitive function is expressed by different cases in North Saami (locative), Lule Saami (inessive) and South Saami (genitive). Nevertheless, `@HAB-tag` is assigned to all of them. The integration of the different disambiguation rule sets is presented in (Antonssen et al, 2010).

The mapping of dependency tags is done in a Constraint Grammar module common to all the Saami languages, and the rule set is compiled with the Visl CG3 compiler ((visl, 2008)) On the dependency level, syntactic tags for verbs are substituted

[1] `http://giellatekno.uit.no/doc/lang/sme/docu-sme-syntaxtags.html`

by other tags (according to clause-type) in order to make it easier to annotate dependency across clauses.[2]

4.1 Disambiguation

In order to test the disambiguator, we took a South Saami corpus of 142.500 words (55% Bible texts and 45% administrative texts). Our South Saami morphological analyser accepts substandard lemma and inflection forms. For frequent typographical errors we have a correction procedure. Despite of this, 12.395 words, or 8,7% of the corpus, were not recognized by our morphological analyser. The unknown words are partly due to the immature status of our morphological analyser, and partly due to the high degree of errors and non-normative forms in South Saami texts. The texts in the corpus were written at a time when there was no spellchecker available for South Saami. The written norm is new and unstable, and rules for writing loanwords are not established. The texts also contain upper cased headlines, which the analyser is not able to analyse, and there are proper nouns and some Norwegian words, which are not recognized by the analyser.

We made two versions of the corpus, one where the unknown words were removed, and one where all the sentences containing at least one unknown word were removed. Unknown words are uninteresting for disambiguation, with no analysis they trivially have no ambiguous cohorts either. Sentences with unknown words are also problematic, since the unknown words may influence upon the analysis of the remaining sentence. In order to look at disambiguation of sentences without unanalysed words, we removed all sentences with unknown words. In our test corpus, we have a missing rate of 8.7% words, and by removing all the affected sentences we lose 64% of the corpus. We are therefore also interested in looking at to what extent the unknown words influence the lemmatising.

The results may be seen in Table 2. The table shows the results for the whole corpus (left column), for the whole corpus analysed with a guesser (central column) and the subcorpus with fully analysed sentences (right column). For each corpus is shown the degree of homonymy (analyses per 1000 words) before and after disambiguation. We then show the result for lemma + PoS (lemmatising), first for all PoS, and then for a reduced PoS set, containing just 4 PoS's (N, V, A, other).

The results improve as we reduce the level of precision, from full analysis, PoS only, to a reduced 4-membered PoS set. For many lemmatisation purposes, distinguishing between different closed classes is not that interesting, and the relevant level of disambiguation is thus 1.056-1.058.

Surprisingly enough, the results for disambiguating the whole corpus is slightly better than the results for disambiguation of the corpus containing fully analysed sentences only. The reason for this is probably that a very large part of the remaining corpus is the Bible, which contains very few words unknown to the analyser, but which has a syntax more demanding for the disambiguator. The administrative texts contain many unknown words, but they are characterized by a more monotone syntax.

We have also tried to improve the result for the specific gold corpus with a word guesser for the unknown words. The word guesser is made with CG, and gives POS and morphosyntactic analysis of the word in question, based upon the word coda. The mid column in Table 2 shows the results of an analysis of the full corpus, where the analysis phase is proceeded by the word guesser. This information is then given as part of the input to the disambiguator. After the disambiguation phase the guessed readings were conflated to one. As can be seen from the table, the guesser component did not give rise to improved results, on the contrary, we see a slight decrease, as compared to the analysis without a guesser.

The main reason for that is this the disambiguator is still in an initial state, where the bulk of the rules are targeted at specific lemma pairs. When input from the morphological guesser is introduced, the picture is completely altered. Now, homonymy across PoS classes is the rule, and not the exception. The disambiguation rules are not written to handle this situation, and the guesser does not improve the results.

Another weakness of the guesser is that it at present gives suggestions on the basis of coda shape only. In a future version, we will add conditional tests to the guesser, and give suggestions based upon syntactic context as well.

[2] http://giellatekno.uit.no/doc/lang/common/docu-deptags.html

Analysers	Languages		
lexicon and morphology	North Saami analyser	Lule Saami analyser	South Saami analyser
disambiguation	North Saami disambiguation	Lule Saami disambiguation	–
syntatic functions	common Saami analyser		
dependency	common Saami analyser		

Table 1: The common Saami analyser infrastructure. The disambiguation of South Saami is the missing link.

Table 2: Homonymy in South Saami

	Whole corpus 8,7% unkn wrds	Whole corpus with guesser	Fully analysed sentences only
Number of words	218.574	218.574	83.530
Analyses per thousand words			
Analyses with homonymy	1.633	1.633	1.792
Present disambiguation	1.112	1.192	1.248
Lemma + PoS disambuguation	1.061	1.141	1.063
Lemma + PoS disambuguation without distinguishing closed PoS	1.056	1.136	1.058

4.2 Precision and recall

For evaluating the accuracy of the disambiguator, we have used two gold standard corpora.

The general gold corpus is a small balanced corpus containing 100 sentences (30 sentences from the Bible, 30 sentences from fictive texts and 40 sentences from newspapers, altogether 1301 words).

The specific gold corpus is closer to the kind of texts, which the disambiguator is meant for. It is an unknown corpus containing 2329 words, 6,7% of them are unknown for our fst. The corpus contains parts from two texts which could be interesting for extracting terminology – one is the *Convention on the Rights of the Child*, and the other one is from a school curriculum about reindeer herding. The results of the analyses are presented in Table 3.

Looking at the results, the disambiguator has a very good recall, as good as 0.98 for full disambiguation and 0.99 for POW disambiguation. As it stands, the program is thus very careful, to the degree that it almost does not remove correct readings. For full morphosyntactic disambiguation , the precision is lower, 0.87 and 0.88, these are poor results in a CG context. Partly, this is the results of some syntactic idiosyncrasies in our special test corpus. But above all it reflects the immature status of the disambiguator. With only 115 disambiguation rules, compared to the 2-3000 rules usually found in standard CG grammars, 0.87 is a good starting point.

For the task at hand, lemmatisation and POS marking, the precision results are much better, 0.93 and 0.94, respectively. Despite the low number of rules, they are efficient enough to carry out POS disambiguation. The remaining degree of homonymy reported for lemma + POS in Table 2 (1.06) thus comes with a precision and recall of 0.94 and 0.99, respectively.

We tried to improve the disambiguation of the known words, by getting more context for the CG-rules in the disambiguator with help of a word guesser. The testing shows however that giving word guesser analysis to the unknown words, does not improve the disambiguation for the known words.

4.3 Discussion

A full fledged constraint grammar typically contains several thousand rules. The South Saami disambiguator is still in an embryonic state, and contains only 115 rules. With this small

Table 3: Precision and recall

	Special gold corpus		General gold corpus	
Number of words	2329		1301	
Unknown words	6,7%		0	
	Prec	Rec	Prec	Rec
Lemma + full disambiguation	0.876	0.980	0.884	0.968
Lemma + PoS disambuguation	0.939	0.990	0.938	0.981
Lemma + open PoS disambuguation	0.945	0.992	0.994	0.987
Lemma + full disambiguation w/guess	0.877	0.978	-	-
Lemma + PoS disambuguation w/guess	0.940	0.988	-	-
Lemma + open PoS disambuguation w/guess	0.947	0.991	-	-

rule set, we are still able to disambiguate text down to 1.100 lemma + PoS readings per 1000 word forms. The rules were written with full grammatical disambiguation in mind, and a rule set geared towards lemmatisation only could have been made even smaller. Figure 3 shows the cumulative effect of the CG rules. The 20 most efficient rules account for almost 80% of the disambiguation.

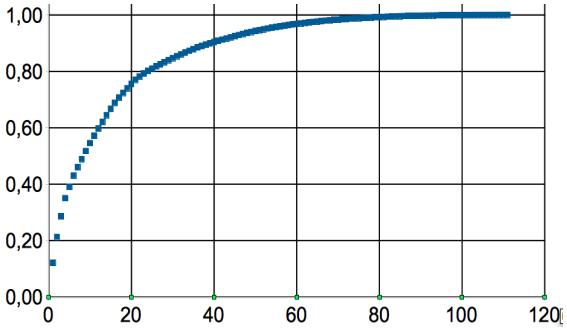


Figure 3: Rule coverage (x = number of rules, y = coverage)

The 10 most efficient CG rules are listed below. For each rule, only the action (select or remove readings) and scope (POS, grammatical feature or lemma), is given. In addition, each rule contains conditional tests for the action in question. For the sake of brevity, these conditions are not given here.

1. IFF: ConNeg if Neg to the left
2. SELECT: Inf is V to the left selects Inf
3. SELECT: A Attr if NP-internal N to the right
4. REMOVE: Imprt if not domain-initial
5. IFF: *goh* is Pcle if in Wackernagel position
6. SELECT: Po, not Pr, if Gen to the left
7. REMOVE: Prefer lexicalised verb to derived
8. REMOVE: *ij* is Periphrastic Neg Prt only if 2nd part of it is present
9. REMOVE: Prefer lexicalised passive to derived
10. REMOVE: Prefer Pers to Dem if no NP-internal N/A/Num to the right

As shown above, the most efficient rules are rules for distinguishing closed PoS. This disambiguation is useful for the rules made for disambiguating open PoS with different lemmas.

Looking now at lexical disambiguation, the 10 most efficient rules for distinguishing between lemmas in open PoS are listed below. The actual word form is given in Italic.

1. SELECT: *Jupmele* – Prefer N Prop to N
2. REMOVE: *Dan* – Prefer Pron Pers to Prop
3. REMOVE: *tjïrrh* – Prefer Po to V
4. REMOVE: Prefer *almetje* N to *elmie* N
5. REMOVE: Prefer *almetje* N to *alma* N
6. REMOVE: Prefer *gïele* N to *gïeledh* V
7. REMOVE: Prefer Adv to A
8. IFF: Interj or other PoS
9. REMOVE: *tjïrrh* – Prefer Po to N
10. SELECT: Prefer V not N

Most of these rules are made specifically for the most frequent lemma pairs having homonym inflectional forms. One improvement strategy might be to make these rules more general and lemma-independent, thereby targeting other lemma-pairs as well.

After disambiguation, there remain 5632 ambiguous word forms, 27.5% of them have the same PoS, and 32.0% of them have the same lemma, as shown in Table 4.

Table 4: Remaining homonymies

	Number of analyses	Percentage
Homonomy with same PoS	1551	27.5%
Homonomy with same lemma	1797	32.0%
Total	5632	100%

The remaining homonymies are mainly of the following types:

1. The same lemma, but different PoS, eg. *juktie* N ('carcass') vs. *juktie* CS ('so that').
2. Different lemmas and different PoS, eg. *vihte* N ('wit') vs. *vihth* Adv ('again').
3. Different lemmas, same PoS and inflection eg. *båetedh* V ('to come') vs. *böötedh* V ('to mend, to pay a fine'). These are the really hard ones to disambiguate.
4. Different lemma, same PoS, but inflection is different (one of them may be derived from the other), eg. *utniedidh* V ('to held') vs. *utnedh* V ('to have, to use').
5. The same lemma has one reading as Proper noun and one as common noun – *Saemie* N ('Saami') vs. *saemie* N ('saami').
6. There are two orthographic variants of the same lemma, which should have been subsumed under the same lemma, eg. *ussjiedidh* V vs. *ussjedidh* V ('think').
7. Derivation vs. lexicalisation, eg. like for *ryöjnesjæjja* N vs. *ryöjnesjidh+V+TV+Der1 +Der/NomAg+N* ('shepherd').

The three first types are true instances of homonymy, many of them can only be resolved by lemma specific rules. The fourth type may or may not be resolved, dependent upon the task at hand. The fifth type is found in some very frequent lemmata. In many instances, this distinction is irrelevant and should be ignored, in other instances one might want to disambiguate them. The last two types are irrelevant for any semantic purposes.

Figure 4 shows the cumulative homonymy for word forms not assigned to a single lemma. Some word forms are very frequent, and writing word specific disambiguation rules for, say, the 50 most common words will already reduce the remaining homonymy with one third.

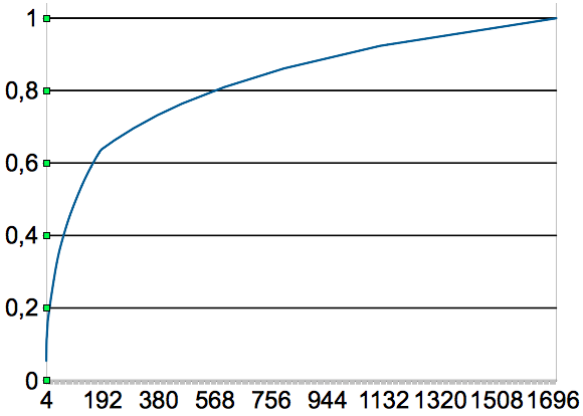


Figure 4: Cumulative homonymy (x = word forms, y = homonymy)

5 Conclusion

The paper has shown that even a tiny set of 115 disambiguation rules is able to achieve quite good results for lemmatising and POS tagging, with a disambiguation rate down at 1.06. In order to disambiguate the full grammatical analysis, a more thorough disambiguation is needed, here the results are about 1.12 even if the corpus contains unknown words. A word guesser doesn't improve the results particularly.

The results also show that the constraint grammar formalism is robust against badly analysed morphological input. As a matter of fact, it scores slightly better on a corpus with an 8.7% error rate, than on a perfect corpus. Even though the difference is probably due to systematic differences in the corpora themselves, it at least shows that constraint grammar is a robust framework for syntactic analysis, capable of dealing with noisy data.

- A small-size CG (115 rules) gives an accuracy of 1.118 - 1.058 readings/word.
- 1/6 of the rule set removes 80% of the homonymy.

- The CG is robust enough to give good disambiguation even with an fst coverage of only 91.3%.
- Adding the results from a morphological guesser did not improve the disambiguation results. More work is needed in order to make use of guesser input.
- The disambiguator's recall is very good, 98.0%. Precision is lower, 87.6-88.6%, and the main focus for improving the South Saami disambiguator will be to improve precision.
- The rule set is a good starting point for a full-fledged disambiguator.

The general conclusion is that even a small-size constraint grammar is able to provide results good enough for POS tagging, lemmatisation, and several other purposes. In order to get a syntactic analysis at the level achieved by other constraint grammars, more work is needed.

References

Lene Antonsen, Trond Trosterud and Linda Wiechetek. 2010. Reusing Grammatical Resources for New Languages *Proceedings of the LREC*. Association for Computational Linguistics, 2782–2789, http://www.lrec-conf.org/proceedings/lrec2010/pdf/254_Paper.pdf

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

Pekka Sammallahti. 1998. *The Saami Languages: an Introduction*. Davvi Girji, Kárášjohka.

VISL-group. 2008. *Constraint Grammar*. http://beta.visl.sdu.dk/constraint_grammar.html Institute of Language and Communication (ISK), University of Southern Denmark

WikiTrans: The English Wikipedia in Esperanto

Eckhard Bick
GrammarSoft ApS & University of Southern Denmark
eckhard.bick@mail.dk

Abstract:
WikiTrans is a translation project and web portal for translated Wikipedias. Using the GrammarSoft's rule-based GramTrans technology, we created a high-quality English-Esperanto machine translation system, and used it to translate the entire English Wikipedia (ca. 3.000.000 articles), at a speed of 17.000 articles a day. The translated articles are searchable both locally (www.wikitrans.net) and in the original Esperanto Wikipedia, where we maintain a revision interface for users who wish to turn our translated articles into new "originals". In this paper, we explain the framework and challenges of the project, and show how translation rules can exploit grammatical information provided by a Constraint Grammar parser.

1 Motivation

In practice, Wikipedia is now the world's main encyclopedic information source, both in terms of size and user base, and although the quality of individual articles may vary, a system of mutual author control, sourcing enforcement and dispute or excellence markers help users to judge the quality and trustworthiness of a given article. However, in spite of being egalitarian and democratic from an authoring point of view, Wikipedia is far from balanced language-wise. Thus, its English information content is considerably larger than that of other languages and completely dwarfs that of minor languages (Fig. 1). The difference is visible not only in the amount of head words covered, but also in the depth and research level of the individual article. In a sense, language barriers are preventing Wikipedia from achieving its primary goal - to make the knowledge of the world accessible to all its citizens..

The Esperanto Wikipedia, although impressive in relative terms, compared to the size of its user base, and as large as e.g. the Danish one, has only 140.000 articles, while the English Wikipedia with its 3.4 million articles (or 2.345.000.000 words) is roughly 24 times as big. In addition, there is a difference in article size[1], with an average of 3.600 letters (~ 600 words) for English and German, and a little over 1500 letters (~ 250 words) in Esperanto, translating into an even bigger factor of difference, 57, when focusing on content volume. In other words, more than 98% of the English language information is not accessible in Esperanto (or Danish). One could argue that the Esperanto articles concentrate on the important and frequently sought-after topics, but it is not least in this kind of major articles that the difference in depth is most palpable, compounded by correspondingly fewer internal links (indirect depth shortage).

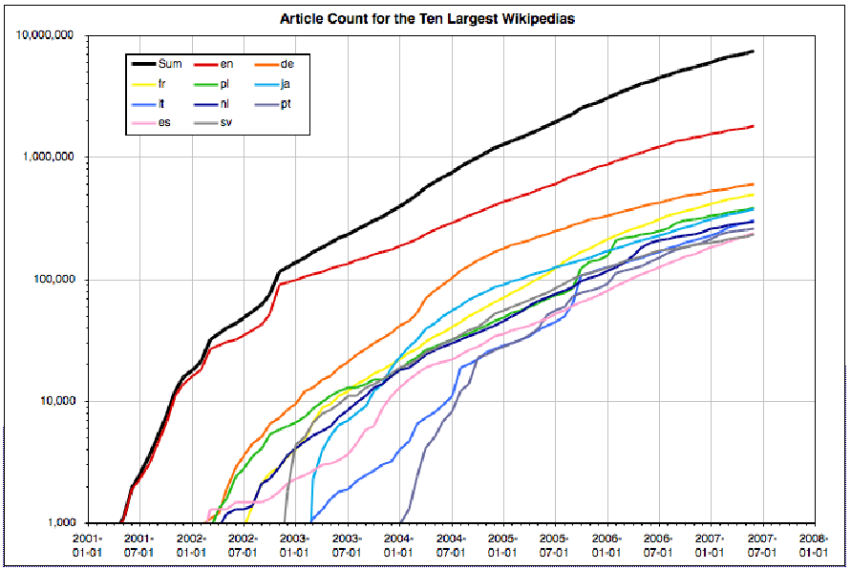


Fig. 1: Chronological language statistics for Wikipedia

Even at the price of some cultural biasing, one obvious solution for this problem is the translation of the English Wikipedia into Esperanto, thus permitting Esperanto readers from different

1 http://stats.wikimedia.org/EN/TablesArticlesBytesPerArticle.htm

countries to access the English ”über-Wikipedia”, and possibly those in other major languages (as defined by size of articles, culture or number of speakers). Manually, at a translation speed of 500 words an hour, such an English-Esperanto translation would cost 4.690.000 man hours. In Denmark, this is equivalent to 3.000 man years, or - at 0.25 EUR/word - ~ 600 million EUR. An unimaginably large sum, beyond any hope of public, let alone private or commercial funding. And even if a one-time funding could be found, it would not be possible to maintain translations in sync with originals, resulting in a rigid system difficult to update.

2 Our solution

The only logical solution to this dilemma, in our view, is the use of machine translation (MT) to save man power, possibly in combination with voluntary linguistic post-revision, for instance concerning major topics, or simply motivated by user interest, professional or private. MT is capable of solving both the quantity and the updating issues, because it allows easy and regular addition of new articles or the management of changes in existing articles. A possible problem for an MT solution is the fact that Wikipedia articles are by no means simple texts, that the lexicon covered is gigantic in its encyclopedic nature, and that any serious user community would demand a fluent and accessible translation without too many errors or untranslated source-language inserts. For the majority of languages there simply is no MT system of sufficient quality, and Esperanto, in particular, is virtually absent from the inventory of the usual commercial MT providers, be it Google, Systran or others.

Technically, MT falls into two technological camps - on the one hand rule based, symbolic systems, on the other statistical machine-learning systems, both having advantages and disadvantages. The traditional solution is the rule-based one, in line with the analytical-structural tradition of general linguistics. The method is, however, very labor-intensive, and too dependent on specialized linguistic skills to be of interest to commercial companies, if the language in question is small in market-economic terms. Statistical MT (SMT) does not need linguists and authors, but only their data, and with a bilingual text collection (a parallel corpus) and preferably as linguistically annotated text data, it is possible to cheaply train a translation model for a new language or domain. In this approach, the problem is that quality is proportional to the amount and quality of training data, and that good SMT therefore needs huge human-translated, i.e. parallel, corpora. Google, for instance, has this in the form of people's bilingual web pages, but not in sufficient quantities for small languages.

GramTrans (Bick 2007-1) is a relatively new approach to MT. Though rule based, the system saves some of the work by exploiting the robustness and depth of existing Constraint Grammar (CG) analyzers (Karlsson 1990). Mature CG parsers offer both better coverage and higher accuracy than most systems, so that GramTrans can build on the linguistic information already available in syntactic-semantic CG analyses of a given sentence (Fig. 2). For instance, the translation module can exploit dependency links between words, as well as their function tags (e.g. 'subject', 'predicative') and semantic classes (e.g. 'tool', 'vehicle', 'food'), in order to craft conditions for the selection of one or other translation alternative in the case of ambiguous constructions, polysemous words, or usage-governed synonym conventions. While CG rules remove, select, add or change linguistic tags (PoS, inflexion, function ...), translations rules simply add yet another layer to this process, targeting translation equivalents and movement operations rather than tags. In operational terms, GramTrans' MT rules are very close to CG proper, since both types of rules work by checking a list of context conditions (e.g. neighboring or dependency related words and their functions or semantic types, valency fillers etc.).

Traditional Constraint Grammar is designed to work on raw, running text, with linguistic analysis and corpus annotation in mind. While most systems do handle sentence separation, tokenization and abbreviations fairly well, and some are robust enough to manage simple corpus markup, they will not automatically handle full xml, documents or the like. In an applicational context, not least when working on heavily layouted text such as Wikipedia, with images, tables, footnotes, links and macros, wrapper solutions are therefore necessary. In order to separate layouting information from grammatical information, we implemented a system where all such information is turned into so-called style tags. This solution permits the wrapper program to reconstitute the exact text attributes and layouting after the CG and translation steps, while at the same time allowing CG rules to make active disambiguation use of such non-linguistic information, for instance in order to recognize titles or links as linguistic units deserving separate syntactic treatment.

3 The WikiTrans project

GramTrans is the motor in the MT technology used by the Danish company GrammarSoft, which offers,

in cooperation with the Norwegian company Kaldera, translations between the Scandinavian languages, and between these and English. GrammarSoft has a close cooperation with the University of Southern Denmark, and a correspondingly strong focus on research, so it was possible to launch WikiTrans, a project without any obvious commercial potential, with the explicit goal of making major language Wikipedias accessible to minor languages, with the English-Esperanto language pair as a proof of concept. Apart from the author, also GrammarSoft's programmer, Tino Didriksen, has been involved in the project.

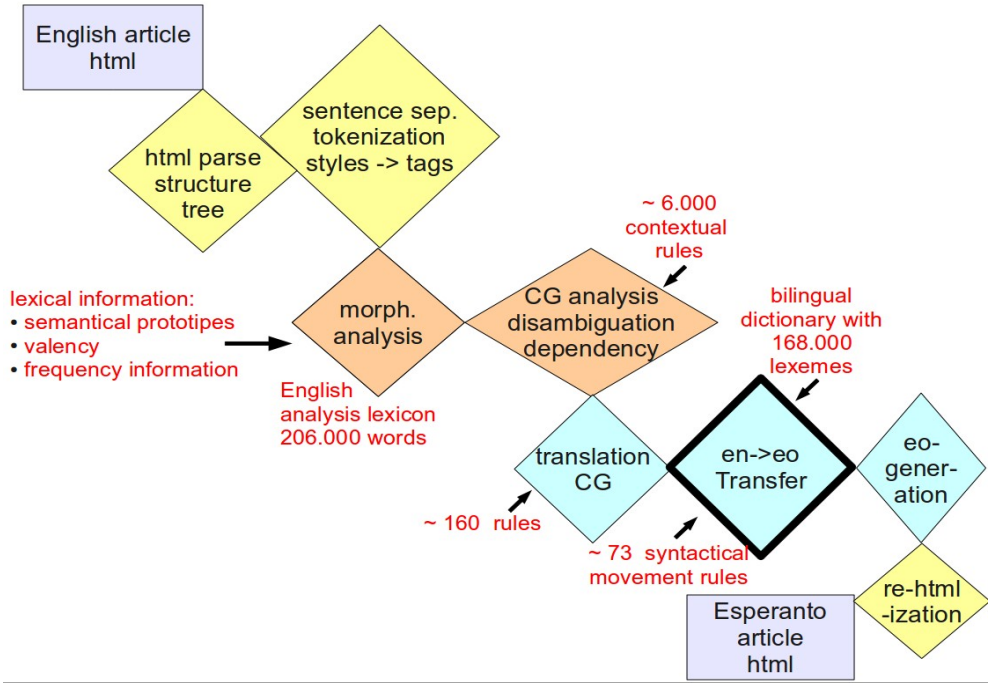


Fig. 2: Flow chart of the WikiTrans modules

The WikiTrans-project was conceived in 2009, and has gone through the following phases:

- preparation phase: 2009 - February 2010: linguistic and lexicographic work
- 1st translation phase (Feb/Mar 2010): 100.000 most frequently read articles
- 2nd translation phase (Mar-Jun 2010): 500.000 longest articles, plus articles with one-word titles (i.e. items more likely to be nouns than names)
- 3rd translation phase (Jun-Dec 2010): the main bulk, ultimately covering all 3 million articles
- use phase: updating, re-translations, human revision

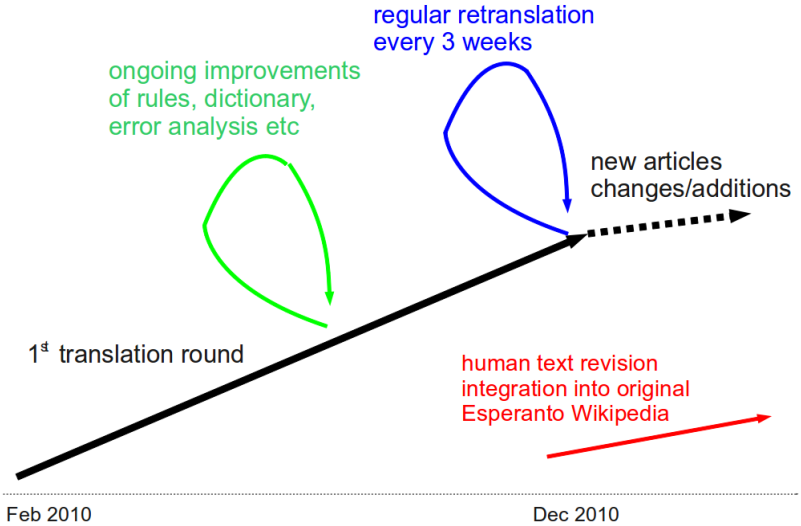


Fig. 3: Project phases of WikiTrans

4 The search interface of WikiTrans

An important reason for translating the whole Wikipedia, rather than simply translate the individual article once a user asks for it, is the possibility to systematically access and search all information. Live translation, though technically possible, would mean either searching in English or translating the search term into English, then choosing between the (English) articles before translating one of them live. Such a service would in reality only serve near-bilingual users preferring to

read in Esperanto rather than English. To really search in another language, the to-be-searched text has to exist in that language, especially considering that many search terms may not even be title words themselves, but still occur several times within the text body of different articles. On the technical side, pretranslated articles allow faster loading, and smoother internal link navigation, and allow a separation, and therefore optimization, of translation infrastructure and server search load.

For WikiTrans, we use the open-source search program *Lucene,* which allows multiple search terms at the same time, and contains an algorithm to order hits according to probability and relevance, based on term frequency and co-occurrence in individual articles. *Lucene* marks this with a probability index between 0 and 1, to which we have added a few further criteria: For instance, a article will be moved to the top of the list if the search term appears as part of the title, or - in the case of a multi-word search expression - if the words appear next to each other, overriding in-article frequency counts. The user is presented with a list of max. 20 hits, providing both title and a short snippet (Fig. 4) to allow quick, but informed selection clicks. The chosen article or articles will be presented with exactly the same layout as the original, with the same pictures, table structure etc., but entirely in Esperanto.

From a technical, programmer's point of view, a very challenging aspect of the search interface was the enormous amount of data - more than 20 GB of text (100 GB with grammatical tags). In order to effectively search a data space of this order, special database optimizations are necessary, and even using cash memory is problematic because at some point searching the growing cash memory becomes less effective than searching the database itself. Unlike a corpus linguist, who is prepared to wait for minutes for the results of a statistical or concordance search, the patience-horizon of the average Wikpedia user is only a few seconds, preferably less than one second. After that, many people may even repress the search button, forcing the server to search for the same information twice, and possibly contributing to server overload.

Fig. 4: From search term to WikiTrans article

In order to allow alphabetic searches, or get an overview over the range of articles, we have also made it possible to simply thumb through the article list from A to Z, using a letter tree ordering system, where the user moves from first to second to third letter and so on, until finally choosing from a one-screen subsection of article names.

5 Links and Bibliography

An important aspect of an electronical encyclopedia, and one of its major advantages over a paper-based one, are internal links. It is such links that combine the readability and fluency of an overview article with the much greater depth of a major background article. Simple back-and-forth clicking will allow everybody to read the article at exactly their individual knowledge level, using or not using internal links to define scientific terms, visualize personal names or explore the thematic context of a given assertion.

Technically, internal links posed several problems: First, during the translation run, there was no guarantee that the linked article had already been translated, so we had to add the (interim) option of live translation, and make sure that sufficient server capacity was available. Second, because the system is handling translations in a semi-intelligent, context-dependent way, the same word chain may receive different translations in different places, with the risk of the translated (in-context) link not matching the (out-of-context) translation of the linked title. We solved this problem by conserving the original English term (or a digital hash representation of it) in the <a href> mark itself, invisible to the user. After the translation and database creation phases, we then in a third step (taking almost a week) matched link translations to title translations.

External links and references are technically more simple, but often full of names, abbreviations and numerical expressions making translation difficult. After first trying to translate as much as possible, we now apply a more cautious policy, not translating a large part of the names, and discussing the option of not translating book and film titles either. Because it is difficult for an automatic system to be reasonably sure what is a work of art, personal name, publisher name or town name, the simplest solution would be not to touch Wikipedia bibliography sections at all, not least considering that the external sources linked will themselves not be in Esperanto, and in a certain sense often serve the function of authenticity proof more than that of background reading.

6 Integration with the monolingual Esperanto Wikipedia

The feedback reactions WikiTrans has received from the Esperanto community, were generally very positive, though many seemed to focus on the publicity aspect more than on the information content aspect. It is difficult for a lay person to appreciate the difficulty of the task, or to compare results with those for other minor languages in Google's translator, Babelfish or the like, and - understandably - the most common critical comment was therefor that translation quality was not good enough, and that the project might ”dilute” the quality of the existing Esperanto Wikipedia. And of course, though good enough for fluent reading, our automatic translations are by no means error-free, nor is a translated article a new original.

Still, this argument can be refuted by pointing out that even without an MT system, it has always been the case that minor-language Wikipedia authors have heavily borrowed from articles in other, major languages by means of translation. In fact, the open-source framework of Wikipedia encourages and supports this flow of text from one language to another. Is it not then better to perform this work more efficiently and rapidly with the help of an automated system? What is needed, is simply marking what's what, and where the user is in a browser clicking chain at any given point in time. Our own proposal is a traffic light colour marking - a red corner mark for a ”virgin” MT-derived WikiTrans article, green for a fully revised article and yellow for a superficially revised article. ”Green” articles could then be moved into the ”true” Wikipedia (while retaining the marker), and red or yellow articles would be searchable both through the WikiTrans portal and - in the case of search failures, or to increase accessible information - in the monolingual Esperanto Wikipedia itself. Fig. 5 shows our scheme for integrating translated and original Wikipedias.

In consultation with Wikipedia administrators, we addressed the practical aspects of this integration between July 2010 and February 2011. The current state of affairs is a solution where user-side javascript programs interact with the GramTrans software at its own server. The user-side software was developed by Marek Blahus (E@I), while Tino Didriksen (GrammarSoft) implemented the necessary GramTrans interface, handling the slightly idiosyncratic internal Wikipedia-syntax, and creating a graphical revision interface. At the time of writing it is already possible for individual registered Wikipedia users to activate the revision-

and-integration module, and parallel WikiTrans searches have been activated for the general public, using WikiTrans as a fall-back solution for search failures.

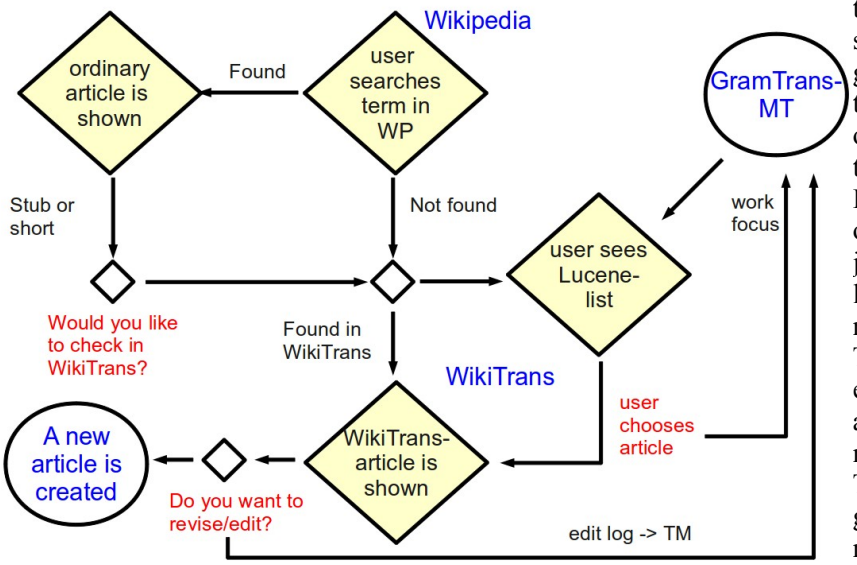


Fig. 5: Integration with the original Wikipedia

7 Linguistic aspects of the translation interface

From a classification point of view, GramTrans is neither a surface MT system nor an interlingua system (Fig. 6). It avoids, of course, the problems of simple word-for-word translations, but does not risk abstraction all the way up to an interlingua level. The ”costs”, in terms of robustness losses, for a full symbolic interlingua are very high, and it is possible to achieve the same with a somewhat ”flatter” transfer from source to target language - simply because most language pairs have more in common, structurally and semantically, than there are differences. This is true also for the English-Esperanto language pair - even more so, because Esperanto with its constructional flexibility is an ideal target language, allowing to mold translations to grammatical patterns found in many different languages without the results sounding unnatural.

As pointed out above, GramTrans relies on comprehensive and robust analysis of the source language, in this case provided by the EngGram parser (http://visl.sdu.dk/visl2/constraint_grammar.html). EngGram is a CG system with more than 6000 rules, a 200.000 word core lexicon, and a dependency style syntactic analysis (Bick 2005). In experiments reported in (Bick 2009), EngGram was evaluated on Wikipedia texts with F-scores of 98.2 and 93.4 for PoS/morphology and syntactic functions, respectively. GramTrans exploits the categories and word links from the EngGram source language analysis in order to create lexical transfer rules designed to resolve semantic ambiguities and choose the correct translation equivalent among several options. The third step, generation, profits heavily from the morphosyntactic flexibility of Esperanto, and from the fact that the generation of an Esperanto morpheme (ending or affix) is almost equivalent to just specifying the desired linguistic category (tense, number, part of speech etc.). The task is made almost embarrassingly simple by the almost perfect regularity and modularity of the language. The only complication in generation is therefor syntax, or rather word order, because in spite of an officially free word order, Esperanto does of course have fairly strong usage conventions with regard to constituent order, and ignoring them - even if not agrammatical as such - would impair fluent reading.

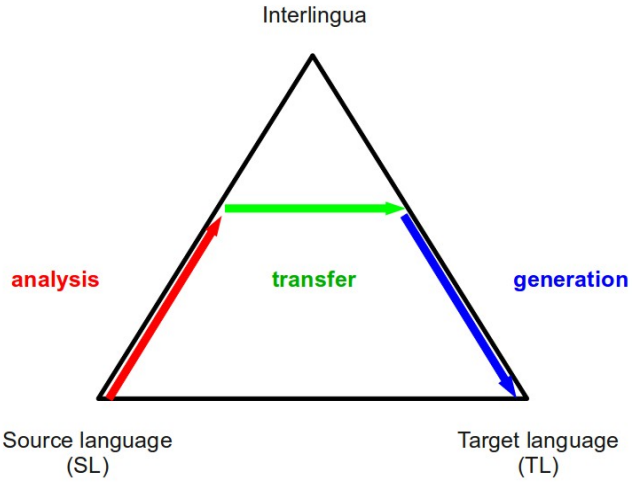


Fig. 6:The translation triangle

7.1. Lexical transfer

The simplest way to exploit Constraint Grammar tags for lexical transfer is one-dimensional in the sense that only local tags (i.e. of the word itself) are used as sense discriminators. This method simply exploits part of speech (1-2) or inflexion (3-4):

1. type_N (noun) **:tipo, :speco**
2. type_V (verb) **:tajpi**
3. force_NS (singular) **:forto**

4. force_NP (plural) **:armeo, :trupo**

In a two-dimensional approach, transfer choices are based on contextual CG information instead, either directly, or indirectly in the form of local tags with relational meaning, such as function tags (5), semantic roles or valency instantiation (e.g. <¤vr> for reflexive verbs where a reflexive pronoun has been found in the context.

5. rather_ADV ... S=(@ADVL) :prefere;
   S=(@>A) :sufiĉe;

Even lexeme-bound traits such as morphological features or semantic class can sometimes be harvested from context, as when nominal agreement features are propagated from head noun to (underspecified) determiner or attribute. An example from the generation task is the fact that Esperanto adjectives have number, while English ones don't, and we use CG propagation rules to add the correct number feature to adjectives. And in the lexical transfer module the ±human is frequently exploited as a translation discriminator, and can be contextually propagated by projecting the feature onto nouns that function as subjects of cognitive or communication verbs, even if the noun itself is sense ambiguous or semantically unmarked due to incomplete lexical information.

6. too_ADV ... S=(@ADVL) :**ankaŭ**;
   S=(@>A) P2?=(INFM)_por :**tro**;
   D=(@>A) :**tro**

Example (6) contains both indirect relational tags (function tags for S=self) and direct relational tags (function tags for D=dependent), as well as positional conditions (P2=second word to the right). All in all, our transfer rules use the following relations:

Dependency: S=self, D=daughter, M=mother,
B=brother, GD=granddaughter, GM=grandmother
Position: right P1, P2 ... Pn, left P-1, P-2 ... P-n

The targeted distinctions do not necessarily reflect conventional dictionary or encyclopedic distinctions. Among other things, metaphors or genre-variation may well be isomorphic in the two languages, making an explicit distinction irrelevant. In more general terms, one can say that one of the biggest secrets of MT (and an important reason for not going all the way to the top of the translation triangle) is the importance of *distinguishing* rather than *defining*. In other words, it is sufficient to have enough context and semantic knowledge in the system to select one or other translation equivalent, but the final *understanding* will only occur in the mind of the target language reader, who has more world knowledge and other background context than any computer could possibly have - so there is no need for the system to explicit everything at an abstract, super-linguistic level. A large part of the semantics is simply transported unaltered from source to target language, without real disambiguation having taken place. For instance, the metaphorical use of containers as units works similar in all languages (2 *glasses of* beer - 2 *glasoj da* biero). On the other hand, it may sometimes be necessary to separate (mainly target language) usage-differences (synonyms, frequency considerations), on top of possible sense distinctions. This problem is less pertinent in Esperanto than in other languages, but it does exist.

Together, the various disambiguation techniques permit quite complex lexicographical work, the most important aspect being the possibility to link verbal information with tags attached to the complements of a given verb (Bick 2007-2). The example below shows how contextual transfer discriminators are able to translate the English verb 'apply' into 9 different verbs in Esperanto. Contexts are tried in successive order, and if no later context conditions apply, the first translation is chosen as the default. It is therefor important to make sure that this translation is robust and maximally ambiguous rather than just the most frequent translation for the word in question.

```
apply_V :uzi;
   D=("for")_pri :peti
   D=(<H> @SUBJ) D=("to"'PRP)_por :kandidatiĝi
   D=(@ACC) D=("to" PRP)_al :apliki
   D!=(@ACC) D=("to" PRP)_por :validi
   D=(<(conv|sem)> @SUBJ) D!=(@ACC) :validi
   D=(<(cm.*|rem)> @ACC) :surŝmiri
   D=("dressing" @ACC)_pansaĵo :surmeti
   <¤vr> D=("to" PRP)_pri :koncentriĝi
   D=("match")_alumeto :malestingi
```

[@SUBJ=subject, @ACC=accusative object, PRP=preposition, <H>=human, <conv>=convention, rule, <sem>=semantical, <cm>=concrete mass word, <rem> remedy, substance, <¤vr>=reflexive]

7.2 Multi-word expressions, translations memory and names

In some cases, it doesn't make sense to translate a word chunk analytically-sequentially - the meaning of the whole is not transparent from the meanings of its parts. GramTrans handles these cases as ”words” with internal spaces. The concept covers complex

nouns *(recovery_position - savpozicio),* a very common category in English, but also prepositional or adverbial phrases with summary translations *(in_violation_of - malobee_al, every_inch_as - tute_same, all_year_round - tutjare),* or simply fixed expressions such as *see_also - vidu_ankaŭ.* Multi-word expressions are not only relevant to the translation module, but also play a role during morphosyntactic analysis, where the concept of complex function words, in particular prepositions and conjunctions, simplifies the assignment of syntactic functions and relations: *each_other (unu_la_alian), instead_of (anstataŭ), other_than (krom).*

A similar simplification can be gained from translations memory (TM) lists, common in many MT systems, and useful to cover special words that are always translated in the same way, i.e that are contextually unaffected and that can be inserted into a translation without any need for transfer rules. One field of TM application are terminology lists, which our systems can turn on or off depending on the to-be-translated domain. But it is also possible to use TM to remedy systematic errors, that can be fixed with a once-and-for-all intervention. In the revision interface we programmed for WikiTrans articles, the system thus remembers all human-made corrections. Besides providing an overview of errors and MT feed back, the change log can be fed into a translation memory, or even used to suggest to the reviewer drop-down translation alternatives for frequently mis-translated expressions.

Independently of the name-recognition (NER) qualities of the EngGram parser, names are hard to translate, and being a very productive category, they have an exceptionally bad lexicon coverage. It isn't eve n possible to trust upper case initials, since uppercasing may occur for other reasons, such as sentence-initially, after a colon, or simply as a means of emphasis. Therefore, it is not possible to 100% sure whether a word is a name or an unknown or compound word from another PoS class. From a purely MT perspective, the question is whether to translate a name, retain the original form or transliterate it with target language phonetics. Here, it is important to distinguish between two main scenarios:

(a) institutions and events, to be translated part for part

European Union - Eŭropa Unio,
Olympics - Olimpikoj,
World War II - Dua Mondmilito

(b) personal names and product names, to be left untranslated

George *Bush - **Georgo Arbusto*

For WikiTrans we also have a compromise solution, where the original is retained, but accompanied by a translation in parentheses, for instance in the case of book, music or film titles that are clearly marked as such in Wikipedia's html structures.

8 Generation and Structural transfer

The last step in the translation chain is morphological and syntactic-structural generation. Again, we exploit CG information inherited by the translation module from the EngGram parser. Basically structural transfer is achieved with the help of movement rules that can change the order of constituents (as defined by the set of all dependency daughters of a target word), using CG tag conditions, and optionally adding or replacing grammatical traits or word forms. One of the structural problems we had to solve was turning genitives into (moved) preposition phrases *(Michel's father - la patro de Michael)*. In some cases, no direct translation exists, and only structural rephrasing can approximate the intended meaning, or it may be necessary to add or remove constructions necessary only in one of the languages, such as English *don't* negation, English *do* questions or Esperanto *ĉu*-questions (yes-no-questions).

As suggested above, the second generative task, morphological generation, is very simple in Esperanto, but in cases where Esperanto is grammatically more explicit than English, context may be needed to add the desired feature. Apart from plural agreement on noun dependents, this is the case for the accusative marker *-n,* which in Esperanto attaches to all nominal word classes and had to be recovered from indirect clues such as CG function tags. Also, the two languages differ in their use of participles (e.g. English *have*-tense), and sometimes there are clashes between semantic and surface number *(wages [pl] - salajro [sg], stools [pl] - feko [sg]).*

9 Conclusions and Perspectives

The language technology project WikiTrans (www.wikitrans.net), succeeded in little more than a year to create an English-Esperanto MT system of sufficient quality to automatically translate Wikipedia texts, and finished in December 2010 the translation of the about 3.000.000 articles in the English Wikipedia, at a speed of ~17.0000 articles a day. The system offers not only target language searches inside translated articles, but also allows integration into Wikipedia proper, through a post-

editing interface.

The perspective for 2011 is the creation of a framework for automatical retranslation and updating. For this purpose the project is setting up a linux cluster consisting of 8 four-core computers to handle fast and parallel MT. The hardware has been sponsored by ESF (Esperanto Studies Foundation), and is hosted at the University of Southern Denmark. Depending on the degree in which the community accepts and uses our post-editing interface, we plan regular treatment of error statistics and corrections suggestions.

A remaining linguistic challenge is terminology: Despite the fact that the WikiTrans dictionary with its 168.000 entries is already the largest English-Esperanto dictionary ever produced, many specialized terms continue to be translated using heuristic methods, e.g. analytical or partial translations, transliterations, Latinisms etc. As a minimal goal, these automatic suggestions should be validated by hand (either by the author, or through a community web portal). Also, existing terminological dictionaries should, if copyright allows, be integrated - which is not as easy as it might seem. First, entries that are assumed to be translations, may in reality be explanations, definitions or terms at a slightly different level in the other language, while what is needed is terms that can directly replace a target language term in the same context, with the same inflexion etc. Second, ambiguity may arise between a specialized term and the same word's meaning in everyday language. If such ambiguities are not spotted and handled with transfer discrimination rules, they will result in a deterioration of the system, with rare translations supplanting common ones. Ideally, new terms should be subjected to a discussion in Esperanto professional and scientific communities, stimulating terminological work proper rather as opposed to mere lexicography, but given the size of the language community, for many domains this is not a likely outcome.

Long term, WikiTrans is to cover further language pairs, the 2011 focus being on English-Danish. From a quantitative point of view, this task is similar to Esperanto, both in terms of article number, article size and size of the bilingual MT lexicon, and we therefor expect a certain synergy, for instance in the identification and translation of ”unknown” English complex nouns, and in the harvesting and classification of name expressions. Another logical step would be the addition of another source language for the same target language - Esperanto, which would allow the user to fill in ”cultural information gaps” - a possible problem immanent to any monolingual Wikipedia. A second source language would also make it possible to compare same-topic articles in areas where information may be biased (e.g. politics, history, religion). GramTrans itself already has a working Danish-Esperanto system, and it would be technically feasible to add translations from further languages using open source systems such as Apertium (http://www.apertium.org/), if and when such a system reaches a sufficient quality level.

Bibliography

Bick, Eckhard. 2005. ”Turning Constraint Grammar Data into Running Dependency Treebanks”. In: Civit, Montserrat & Kübler, Sandra & Martí, Ma. Antònia (ed.), *Proceedings of TLT 2005 (4th Workshop on Treebanks and Linguistic Theory, Barcelona, 2005)*, pp.19-27

Bick, Eckhard. 2007-1. ”Dan2eng: Wide-Coverage Danish-English Machine Translation”. In: Bente Maegaard (ed.), *Proceedings of Machine Translation Summit XI, 10-14. Sept. 2007, Copenhagen, Denmark*. pp. 37-43

Bick, Eckhard. 2007-2. ”Fra syntaks til semantik: Polysemiresolution igennem Dependensstrukturer i dansk-engelsk maskinoversættelse”. In: Henrik Jørgensen & Peter Widell (eds.), *Det bedre argument, Festschrift til Ole Togeby på 60-årsdagen* pp.35-52

Bick, Eckhard. 2009. ”Introducing Probabilistic Information in Constraint Grammar Parsing”. In: *Proceedings of Corpus Linguistics 2009, Liverpool, UK*. Electronically published at: ucrel.lancs.ac.uk/publications/cl2009/

Karlsson, Fred. 1990. Constraint Grammar as a Framework for Parsing Running Text. In: Karlgren, Hans (ed.), COLING-90 Helsinki: *Proceedings of the 13th International Conference on Computational Linguistics*, Vol. 3, pp.168-173

Using constraint grammar in the Bangor Autoglosser to disambiguate multilingual spoken text

Kevin Donnelly *and* Margaret Deuchar
ESRC Centre for Research on Bilingualism in Theory and Practice
Prifysgol Bangor University, Wales, UK
{k.donnelly|m.deuchar}@bangor.ac.uk

Abstract

We present a novel use of constraint grammar (CG) in automatic glossing software to disambiguate surface forms in connected multilingual speech. The resulting autoglosser output shows 97-99% accuracy over all three languages. We discuss the CG rules that help deliver this, noting the differences between those applying to Welsh and Spanish, and those applying to English.

1 Introduction

Bangor University's ESRC Centre for Research on Bilingualism,[1] established in Jnauary 2007, has assembled some 130 bilingual conversations in three corpora: **Siarad**[2] (Welsh-English), **Patagonia** (Welsh-Spanish), **Miami** (Spanish-English).

The conversations total some 80 hours and 750,000 words, and are all available under the GNU GPL.[3] Each recording is provided with a detailed transcription in the widely-used CLAN format[4] (MacWhinney, 2000), along with a free translation in English, and an interlinear gloss giving lexemes and part-of-speech (POS) tags for each word, so that researchers without first-hand knowledge of the languages concerned can more easily parse the utterances.

Part of a typical transcription is shown in Figure 1, in which (using CLAN terminology) three "tiers" can be discerned: the speech tier, the gloss tier, and the translation tier.

The speech tier (the words actually uttered) is marked by an initial ID to distinguish the speaker (e.g. **SER*), followed by the transcribed speech (with each word tagged for language[5] – unmarked for Welsh, *@s:eng* for English, *@s:cym&eng* for indeterminate[6]), and two numbers giving the start and end times of the utterance in the audiofile.

	Chats	*Hours*	*Words*	*Date*
Welsh-English	69	40	456k	2009
Welsh-Spanish	32	20	183k	2011
Spanish-English	31	20	126k	2011
	132	**80**	**765k**	

Table 1 – The three ESRC Centre corpora.

The gloss tier is marked by an initial *%gls*, followed by a series of lexeme+POS-tag strings.

The translation tier is marked by an initial *%eng*, and gives a free translation of the speech tier (the speaker's utterance) into English.

The corpora are valuable in examining how language is actually used: for instance, the differences between spoken language and formal written language, sociolinguistic variation (what forms of language are used where and by whom), the balance between languages in bilingual usage, and how one language handles lexical items from the other.[7]

Manual glossing of the Siarad (Welsh-English) proved to be tedious and time-consuming, so in order to save valuable specialist time it was decided to explore automating the glossing of the Miami (Spanish-English) and Patagonia (Welsh-Spanish) corpora.

Although the CLAN project provides a tag-

[1] http://bilingualism.bangor.ac.uk

[2] Siarad means "speak" in Welsh.

[3] http://www.gnu.org/licenses/gpl.html

[4] http://childes.psy.cmu.edu/clan. Note that using CLAN to record bilingual speech is an extension of its original focus on recording language development in children.

[5] The autoglosser handles 4 marking systems, which reflect changes in transcription practice in the ESRC Centre over the past 5 years, and developments in CLAN itself.

[6] Words which are used in both languages, and which therefore cannot be assigned unambiguously to one of them.

[7] For instance, Jon Stammers (Stammers, 2010) has used the Siarad corpus to show that Welsh loan-verbs such as *textio* (to text) behave more like ordinary Welsh verbs the more frequent they are.

*SER: dw i (y)n hopeless@s:eng efo tynnu llun . 72848_73881
%gls: be.1S.PRES PRON.1S PRT hopeless with take.NONFIN picture
%eng: I'm hopeless at drawing
*SER: dw i (y)n tynnu llun i [/] i (y)r plant <i plant> [//] <i (y)r> [//] # i er@s:cym&eng &h Helen@s:cym&eng a Susanna@s:cym&eng a +/. 73881_79477
%gls: be.1S.PRES PRON.1S PRT take.NONFIN picture for for DET children for children for DET for IM Helen and Susanna and
%eng: I draw a picture for ... for the children, for, er, Helen and Susanna and ...

Figure 1 – Excerpt from the file *deuchar1* in the Siarad corpus (Welsh-English).

ging system (MOR),[8] this only caters for 11 languages, each with more than 5m speakers. Vocabulary is distributed over a number of files, and MOR requires a separate pass over the file to tag each language. Post-tagging disambiguation (using the POST program) is only available for 4 languages. Software such as Toolbox[9] offers interlinear glossing capability, but is aimed more at linguistic field researchers, and is less applicable to fully-described languages; moreover, it does not seem to be scriptable, which was essential in order to deal with the volume of data in the corpora.

There appears to be no tagger available at all for Welsh, reflecting the dearth of linguistic tools available to many minority languages (Antonsen et al., 2010).

With no existing software meeting the purpose, a two-week test project in April 2010 looked at the viability of simply writing out entries from Spanish and Welsh dictionaries (see Section 2 below) for each word in the transcription. The results of the tests were encouraging, and the only remaining issue was how to dismbiguate between the returned entries. For this we turned to constraint grammar (Karlsson et al., 1995), and the remainder of this paper reports on how this is used in the autoglossing software developed over the past year.[10]

2 The dictionaries

A key element of any tagging or glossing system is the use of a dictionary to allow lookup of the word in the chosen language.

The Spanish dictionary used in the Autoglosser is based on the one used in Apertium,[11] a free (GPL) platform for developing rule-based machine translation systems. The Welsh dictionary is based on Eurfa,[12] developed by the first author a few years ago, and still the largest free (GPL) dictionary for Welsh. The English dictionary is based on Kevin Atkinson's Moby list.[13]

The use of material with a free or public domain license allows existing lexical resources to be easily adapted and extended for the Autoglosser without having to worry about licensing terms. This is an especially important consideration for minority languages like Welsh,(Streiter et al., 2006) where resources may be limited.

Each dictionary takes the form of one PostgreSQL database table, storing full words (not morphemes). All of the original dictionaries have undergone some refactoring to simplify and standardise their layout, and to correct errors and omissions.[14]

The dictionary table can be easily edited in place, or it can be exported to a CSV file, making it accessible via a spreadsheet for those who are unfamiliar with databases. The dictionary is therefore easy to update, since the format is a familiar glossary-style list of words. This makes expanding or editing the dictionary more accessible for people without extensive computer skills, which is again important for minority languages – no esoteric rules on word-division apply, nor are the contents distributed over several files.

In theory at least, this should simplify the addition of further languages in the future. If a simple wordlist is available, it is possible to plug it into the autoglosser, and get some useful non-disambiguated output immediately; this output can then be progressively refined by the addition of CG rules,[15] and refactoring of the dictionary

[8] http://childes.psy.cmu.edu/morgrams

[9] http://www.sil.org/computing/toolbox

[10] The Bangor Autoglosser software, licensed under the GPL, is available from http://siarad.org.uk/autoglosser.php

[11] http://apertium.org

[12] http://eurfa.org.uk

[13] http://wordlist.sourceforge.net

[14] The English dictionary is particularly prone to include non-existent "words" such as *fam*, *fath*, *gaster*, etc, and further cleaning is still required.

[15] Constraint grammar has been described as "the only grammar-based parser framework" (http://giellatekno.uit.no/cg/11/index.html), and it is indeed very easy for linguists to work with.

lookup to allow a reduction in the size of the dictionaries.

Some entries from the Welsh dictionary are in Table 2. The enlemma column gives the English lexeme for the word, and the pos column gives the part-of-speech (POS).

surface	*lemma*	*enlemma*	*pos*	*gender*	*number*	*tense*
bara	bara	bread	n	m	sg	
cathod	cath	cat	n	f	pl	
mynd	mynd	go	v			infin
aeth	mynd	go	v		3s	past
hapus	hapus	happy	adj			
rhywsut	rhywsut	somehow	adv			
heb	heb	without	prep			

Table 2 – Entries from the Welsh dictionary.

A similar set of entries from the Spanish dictionary is in Table 3 – it can be seen that the same columns are used in both dictionaries.

surface	*lemma*	*enlemma*	*pos*	*gender*	*number*	*tense*
perro	perro	dog	n	m	sg	
canciones	canción	song	n	f	pl	
empezar	empezar	start	v			infin
empieza	empezar	start	v		23s	pres
empieza	empezar	start	v		2s	imper
rojo	rojo	red	adj	m	sg	
rojas	rojo	red	adj	f	pl	
por	por	for	prep			

Table 3 – Entries from the Spanish dictionary.

Both Spanish and Welsh are inflected languages, where the surface forms give clues about the word's POS. English, however, is an analytic language where the POS of the many homophonous words is defined by their role in the sentence. The format for the English dictionary, some entries for which are in Table 4, reflects this by having the POS reflect all of these possibilities, with the correct POS being selected during disambiguation.

surface	*lemma*	*pos*	*number*	*tense*
walk	walk	sv		infin
break	break	sv		infin
broke	break	av		past
broken	break	av		pastpart
car	car	n	sg	
quick	adj			
by	by	prep		
which	which	rel		

Table 4 – Entries from the English dictionary.

For example, **walk** can be a noun (*a short walk*), an imperative verb (*walk the line!*), an infinitive verb (*to walk a mile*) and a present tense verb (*they walk everywhere*). Thus **walk** has the POS **sv**, meaning that it can be either a singular noun or a verb. The main benefit of this approach is that it minimises the number of entries which the dictionary has to include (in this case, one entry instead of four), and therefore makes maintenance of the dictionary easier.

3 The autoglossing process

Each line of the transcribed conversation file is read into an utterances table containing the following fields:

- utterance_id
- filename
- speaker
- surface (the utterance)
- startpoint
- endpoint
- duration
- manual gloss (if present)
- English translation (if present)
- comments (if present)
- precode[16] (if present)

Any non-lexical markers in the utterance are discarded, and it is then split into words, which are stored in a words table with the following fields:

- word_id
- utterance_id
- location of the word in the utterance
- surface (the word)
- automatic gloss (to hold the later output)
- manual gloss (if present)
- language id
- speaker
- filename

Each entry in the words table is looked up against the dictionary table for the appropriate language, using the language assigned to the word by the transcriber.[17]

The lookup includes some basic segmentation of the word. This helps to minimise the number of dictionary entries and make maintenance of the dictionary easier.

For Welsh, the lookup detects mutation[18] and adds corresponding tags:

thad →**tad** (*father*) + am (aspirate mutation)
gael →**cael** (*get*) + sm (soft mutation)

[16]This marks entire utterances in the least-frequent language of the conversation.

[17]In the absence of this, it would in principle be possible to use a brute-force lookup on each dictionary in turn.

[18]Mutation – morphophonemic alteration of initial consonants, which also marks syntactic relations at the clause level – is an important characteristic of the Celtic languages. A Welsh example is: **mae o'n marw** (*he is dying*), but **mae o'n farw** (*he is dead*), where the change **m**→**f** signifies that the mutated word is an adjective and not a verb. These mutations have to be removed in order to get to the underlying lexeme.

For Spanish, tags are added when clitic pronouns attached to verbforms are detected:

ponerle →**poner** (*put*) + **le**[pron.mf.3s]

déjanos →**déja** (*leave*) + **nos**[pron.mf.1p]

For English, tags are added for things like:

(a) elisions:

gonna →**go** # to.prep

we're →**we** # be.v.pres

(b) genitives or verb elisions:

father's →**father** # gb

(c) plural nouns or 3s present tense verbs:

breaks →**break** # pv

(d) adjectives or past tense verbs:

constructed →**construct** # av

(e) adjectives, singular nouns or present participle verbs:

thinking →**think** # asv

(f) adverbs:

quickly →**quick** # adv

All matching entries in the dictionary are then written out to a file in the format required by the constraint grammar parser.[19]

```
"<ddim>"
   "dim"  96,1 [cy] n m sg :nothing: + sm
   "dim"  96,1 [cy] adv :not: + sm
"<yn>"
   "yn"  96,2 [cy] stat :stative:
   "yn"  96,2 [cy] prep :in:
   "gan"  96,2 [cy] prep :with: + sm
"<gynnar>"
   "cynnar"  96,3 [cy] adj :early: + sm
"<iawn>"
   "iawn"  96,4 [cy] adv :OK:
   "iawn"  96,4 [cy] adv :very:
```

Figure 2 – A phrase, after lookup and before disambiguation, meaning "*not very early*", from the file *patagonia1* in the Patagonia corpus (Welsh-Spanish).

```
"<it's>"
   "it"  545,1 [en] pron.sub 3s :it: # gb
"<coming>"
   "come"  545,2 [en] sv infin :come: # asv
"<out>"
   "out"  545,3 [en] adv :out:
"<on>"
   "on"  545,4 [en] prep :on:
"<D_V_D>"
   "D_V_D"  545,5 [en] name
"<then>"
   "then"  545,6 [en] adv :then:
```

Figure 3 – A phrase, after lookup and before disambiguation, from the file *herring7* in the Miami corpus (Spanish-English).

[19] We use the visl-cg3 parser developed by Eckhard Bick and Tino Didriksen - http://beta.visl.sdu.dk/cg3.html

Figures 2 and 3 show the output after lookup of a monolingual phrase in Welsh and English respectively.

The constraint grammar parser applies the rules in the grammar file to discard invalid entries and convert tags where appropriate, and creates another file containing only valid, disambiguated entries. The two phrases given above are shown after disambiguation in Figures 4 and 5.

```
"<ddim>"
   "dim" 96,1 [cy] adv :not: + sm
"<yn>"
   "yn" 96,2 [cy] stat :stative:
"<gynnar>"
   "cynnar" 96,3 [cy] adj :early: + sm
"<iawn>"
   "iawn" 96,4 [cy] adv :very:
```

Figure 4 – The Welsh phrase from Figure 2 after disambiguation.

```
"<it's>"
   "it" 545,1 [en] pron.sub 3s :it: # be.v.3s.pres
"<coming>"
   "come" 545,2 [en] v prespart :come: #
"<out>"
   "out" 545,3 [en] adv :out:
"<on>"
   "on" 545,4 [en] prep :on:
"<D_V_D>"
   "D_V_D" 545,5 [en] name
"<then>"
   "then" 545,6 [en] adv :then:
```

Figure 5 – The English phrase from Figure 3 after disambiguation.

This file is then read into the database, and the glosses (in the form of a lexeme+POS-tag string, following the Leipzig schema (Comrie et al., 2008) so far as possible) are extracted and stored in the words table against each word of the original transcription. At this point, the words table looks like Figure 6, where the words in a Spanish utterance meaning "*And if some lorry goes in there, for example, to leave off furniture or whatever.*" have all been glossed appropriately.

Finally, a text with an interlinear gloss, as in Figure 7, is created by writing out the utterances again, along with the concatenated glosses. Comparing to Figure 1, an additional *%aut* tier has been added for each utterance, in parallel with the pre-existing *%gls* tier provided by manual glossing.

The Autoglosser produces glossed text at a rate of 900-1100 words per minute (depending on

*SER: dw i (y)n hopeless@s:eng efo tynnu llun . %snd:"deuchar1"_72848_73881
%aut: be.V.1S.PRES.SPOKEN I.PRON.1S stative.STAT hopeless.ADJ with.PREP take.V.INFIN picture.N.M.SG
%gls: be.1S.PRES PRON.1S PRT hopeless with take.NONFIN picture
%eng: I'm hopeless at drawing
*SER: dw i (y)n tynnu llun i [/] i (y)r plant <i plant> [//] <i (y)r> [//] # i er@s:cym&eng &h Helen@s:cym&eng a Susanna@s:cym&eng a +/ . %snd:"deuchar1"_73881_79477
%aut: be.V.1S.PRES.SPOKEN I.PRON.1S stative.STAT take.V.INFIN picture.N.M.SG to.PREP to.PREP the.DET.DEF children.N.M.PL to.PREP children.N.M.PL to.PREP the.DET.DEF to.PREP er.IM name and.CONJ name and.CONJ
%gls: be.1S.PRES PRON.1S PRT take.NONFIN picture for for DET children for children for DET for IM Helen and Susanna and
%eng: I draw a picture for...for the children, for, er Helen and Susanna and...

Figure 7 – Autoglossed excerpt from the file *deuchar1* in the Siarad corpus (Welsh-English) – compare Figure 1.

word_id	utterance_id	location	surface	auto	com	speaker	langid
43	7	1	y	and.CONJ		SOF	3
44	7	2	si	if.CONJ		SOF	3
45	7	3	entra	enter.V.2S.IMPER		SOF	3
46	7	4	algún	some.ADJ.M.SG		SOF	3
47	7	5	camión	lorry.N.M.SG		SOF	3
48	7	6	ahí	there.ADV		SOF	3
49	7	7	por	for.PREP		SOF	3
50	7	8	ejemplo	example.N.M.SG		SOF	3
51	7	9	a	to.PREP		SOF	3
52	7	10	dejar	leave.V.INFIN		SOF	3
53	7	11	muebles	furniture.N.M.PL		SOF	3
54	7	12	o	or.CONJ		SOF	3
55	7	13	cualquier	whatever.ADJ.MF.SG		SOF	3
56	7	14	cosa	thing.N.F.SG		SOF	3
57	7	15	.			SOF	999

Figure 6 – An utterance from the words table for the file *sastre1* in the Miami corpus (Spanish-English)

whether the original transcription file already contains a manual gloss tier). The transcription of a half-hour conversation can therefore be glossed in around 6 minutes.[20]

The grammar file currently contains about 500 rules for Welsh, about 200 for English, and around 170 for Spanish. These figures reflect the fact that most work so far has been done on Welsh.

Preliminary results (see Table 5) suggest that the Autoglosser's accuracy is 97-99%, depending on the language.[21] We are confident that the accuracy rate can be further improved.

4 Using constraint grammar

We discuss here two issues:

- The addition of tags in the lookup output to specify language, and the handling of these in the grammar so as to allow one-pass disambiguation of multilingual text.
- The different approaches taken in the grammar to handle the differing nature of the languages (already reflected to some extent in the dictionary entries).

[20]The entire Siarad corpus of around 40 hours duration (456,000 words) was glossed in 8h27m.

[21]A recent comparison (Donnelly et al., 2011) suggests that accuracy is within 2% of manual glossing for Welsh, and comparable to CLAN's own MOR tagger for Spanish.

4.1 Language-specific rules

Multilingual discourse is far more common than has been assumed in classical linguistics, and it is only over the last 20 years that this important area has been given proper attention. The Autoglosser is the first attempt to apply constraint grammar to multilingual text, and in fact only two things need to be done: (1) include the language tag in the output from each word's lookup; (2) put all the rules (grouped according to language for ease of reference) into the same grammar file.

In Figure 8, the phrase oscillates between Welsh and Spanish, and this is reflected in the inclusion of the tags **[cy]** and **[es]** in the readings.

"<mewn>"
"mewn" 128,4 [cy] prep :in:
"<motor>"
"motor" 128,5 [es] n m sg :motor:
"<newydd>"
"newydd" 128,6 [cy] adj :new:
"<internacional>"
"internacional" 128,7 [es] adj mf sg :international:

Figure 8 – A bilingual phrase (*"in a new international car"*) from the file *patagonia2* in the Patagonia corpus (Welsh-Spanish).

In the following noun phrases, the last word (**dro**, **man**, **viaje**) can be both a noun and a verb.
Welsh: **yr ail dro** (*the second time*)
English: **the third man**
Spanish: **el primer viaje** (*the first journey*)
A rule such as:
select (n) if (-1 (ord));
will choose the noun (**n**) reading if the first word to the left (**-1**) is an ordinal (**ord**), meaning that the verb readings for **dro**, **man** and **viaje** will be deleted.

The language tag can be used to constrain the application of the constraint grammar rules to the

	Corpus	*Files*	*Words*	*Accuracy*	*MCL*	*Coverage*
Welsh-Spanish	Patagonia	patagonia1, 2, 3, 6	15,677	99%	W (92%)	100%
Welsh-English	Siarad	stammers4, deuchar1	10,411	98%	W (81%)	96%
Spanish-English	Miami	zeledon5	4,202	97%	S (59%)	97%

Table 5 – Autoglossing accuracy and coverage for sample files from the three ESRC Centre corpora. In the MCL (most common language) column, W=Welsh and S=Spanish. Coverage is 100% for the Patagonia files because all unknown words were added to the dictionaries before autoglossing.

relevant language.[22] Thus, if the above rule is amended to read:

select ([es] n) if (-1 ([es] ord));

it will only apply to the Spanish phrase, and not to the Welsh or English ones, meaning that the verb reading will still be available in those languages.

It is also possible to make the rules apply across language boundaries by selectively removing language constraints.

In Figure 9, the Spanish **otro** can be either an adjective before a noun, or a pronoun. If the selection rule leaves the noun unspecified as to language:

select ([es] adj) if (-1 (ord));

the adjective reading will be selected before any noun (not just Spanish nouns), as in Figure 10.

```
"<es>"
   "ser"  500,1 [es] v 23s pres :be:
"<otro>"
   "otro"  500,2 [es] adj m sg :other:
   "otro"  500,2 [es] pron m sg :other:
"<zip>"
   "zip"  500,3 [en] n sg :zip:
"<code>"
   "code"  500,4 [en] n sg :code:
```

Figure 9 – A bilingual phrase (*"it's a different zipcode"*) from the file *sastre1* in the Miami corpus (Spanish-English).

```
"<es>"
   "ser"  500,1 [es] v 23s pres :be:
"<otro>"
   "otro"  500,2 [es] adj m sg :other:
"<zip>"
   "zip"  500,3 [en] n sg :zip:
"<code>"
   "code"  500,4 [en] n sg :code:
```

Figure 10 – The bilingual phrase from Figure 9 after disambiguation.

In Figure 11, **camping** can be an adjective (*the camping ground*), a singular noun (*camping is fun*), or (as here) a verb. In **vamos camping**, the **asv** tag can be converted to the desired present participle verb tag by referring to the meaning of the preceding verb, so that the rule applies to both English (*go camping*) and Spanish (*vamos camping*):

substitute (sv infin asv) (v prespart) ([en] sv infin asv) (-1 (:go:));

as in Figure 12.

```
"<cada>"
   "cada"  79,5 [es] adj mf sg :every:
"<vez>"
   "vez"  79,6 [es] n f sg :time:
"<que>"
   "que"  79,7 [es] conj :than:
   "que"  79,7 [es] conj :that:
"<nos>"
   "yo"  79,8 [es] pron.obl mf 1p :us:
"<vamos>"
   "ir"  79,9 [es] v 1p pres :go:
"<camping>"
   "camp"  79,10 [en] sv infin :camp: # asv
```

Figure 11 – A bilingual phrase (*"every time that we go camping"*) from the file *sastre1* in the Miami corpus (Spanish-English).

4.2 Tidying readings

The re-use of lexical resources can lead to a conflict – for many purposes, a comprehensive dictionary giving as many entries as possible for a particular word is desirable, but these multiple entries are not required for an application like the autoglosser, where one lemma will usually be sufficient for tagging purposes.

In cases where the entries are archaic or infrequent words, we use CG **select** rules to remove them from consideration. The Welsh words **huno** (*sleep*) and **pallu** (*refuse*) are low-frequency, so the following rules are applied:

remove ("huno" [cy] :sleep:);
remove ("pallu" v :refuse:);

In other cases, where a single word has different meanings we use CG **select** rules to prioritise one of the meanings. The Welsh dictionary gives two

[22] In practice, there is only a small number of cases where full constraint of the rules is essential (because only a couple of dozen words in each language overlap orthographically), but it is prudent at this stage to err on the side of overspecification.

```
"<cada>"
   "cada" 79,5 [es] adj mf sg :every:
"<vez>"
   "vez" 79,6 [es] n f sg :time:
"<que>"
   "que" 79,7 [es] pron.rel :that:
"<nos>"
   "yo" 79,8 [es] pron.obl mf 1p :us:
"<vamos>"
   "ir" 79,9 [es] v 1p pres :go:
"<camping>"
   "camp" 79,10 [en] v prespart :camp: #
```

Figure 12 – The bilingual phrase from Figure 11 after disambiguation.

meanings for **cyfeiriad** (*direction* and *address*) – the following rule ensures that the *address* meaning is ignored:

select ("cyfeiriad" [cy] :direction:);

The lookup process can generate readings which are invalid, and these need to be removed. The cohort of readings for the Welsh word **nos** (*night*) will include an incorrect one interpreting it as a nasally-mutated form of the imperative (**dos)** of the verb **mynd** (*go*), which is linguistically impossible. This sort of entry can be removed with a rule like:

remove ([cy] "mynd" v 2s imper nm);

A similar issue arises when indeterminate words are being looked up. It will be recalled that indeterminate words are those which appear in dictionaries of both languages, so it is impossible to state unequivocally which language they belong to.[23] Since the practice in the transcriptions is to use English spelling for indeterminate words, lookup for these words uses the English dictionary. The interaction with Welsh mutation can lead to invalid readings, such as the interpretation of the hesitation marker **um** as a soft-mutated form of the word gum, which is extremely unlikely. This can be removed with a rule like:

remove ([in] "gum" n sg sm);

4.3 Nature of rules

Spanish and Welsh are inflected languages,[24] while English is an analytic language with few inflections (mainly in "strong" verbs). This is reflected in the nature of the rules that have proved most efficient in the autoglosser.

For Spanish and Welsh, surface forms are fairly well-defined by their shape – **empieza**, for instance, can only be the second/third person singular present or the second person singular imperative of **empezar** (*to begin*). The lookup fetches these entries from the dictionary,[25] and so the rules consist mainly of **select** rules (with a few **remove**s and **substitute**s).

For English, on the other hand, the surface form gives us few clues about the part-of-speech a word belongs to, which is largely defined by its role in the sentence – **break** can be a singular noun, or a verb infinitive, or the non-third person singular present tense. Instead of giving **break** three entries in the English dictionary, we have chosen, as noted in Section 2, to assign it one entry, with a tag (*sv*) which reflects this diversity of role.

The result is that the the vast majority of rules for English are **substitute**s, converting one set of tags into another. For example, the surface word **miniature** can be either an adjective or a singular noun, so it is tagged *as* in the dictionary. Rules such as the following then handle its correct tagging based on context:

substitute (as) (adj) ([en] as) (1 ([en] n) or ([en] pron));

This says that an English *as* tag should be converted to an *adjective* tag when the word is followed by a noun or pronoun (e.g. **a miniature rabbit**, **miniature ones**).

Similar refinement rules can be applied to other parts-of-speech such as pronouns:

substitute (pron.sub) (pron.obj) ([en] pron.sub) (-1 ([en] v infin));

which will correctly tag **it** in **and open it** as an object pronoun, or verbs:

substitute (av past) (v past) ([en] av past) (-1 ([en] pron.sub)) (not -1 (have.v.pres)) (not -2 ("have"));

Here, **bought**, which can either be an adjective (**bought goods**) or a past verb, has the latter selected provided it is not preceded by enclitic or self-standing instances of the auxiliary verb **have**. This correctly tags **we bought**, but passes over **you've bought**, or **we have bought**. These latter examples can be handled by an additional rule converting the tag to a past participle:

[23] This is meant synchronically rather than diachronically, in terms of current usage in both languages – historically, the word may be considered a loanword.

[24] Though it should be noted that in Welsh, particularly spoken Welsh, inflected verbforms are now widely replaced by periphrastic forms.

[25] The possibility of de-conjugating inflected verbs on-the-fly is attractive, but may be too complex to attempt at this stage.

substitute (av past) (v pastpart) ([en] av past) (-1 (have.v.pres) or ("have") or ("be"));

which will also correctly tag **it was bought**.

It can be said that in general these **substitute** rules are more dependent on rule order than **select** or **remove** rules, since the output of a substitution earlier in the stack needs to be taken into account by a rule later in the stack.

4.4 Rule scope

Our current view is that **remove** and **select-if-not** rules are particularly problematic unless they are carefully constrained. A select rule is exclusive in what it applies to, and it might be considered possible to frame a select-if-not rule to be equally exclusive. By its nature, however, the set of negatives is larger than the set of positives, so it is easy to miss something obvious, particularly when dealing with rules that can apply across languages.

An example of this was the results of combining a set of grammar rules for Welsh with a previously working set of rules for Spanish - the result was 304 regressions in the Spanish output. This was traced to a Welsh rule selecting an imperative if the particle **ni** did not appear in first position in the sentence:

select (imper) if (not @1 ("ni"));

Since **ni** did not appear in this context in Spanish, all instances where an imperative reading was possible were selected, giving the regressions. In this case, the rule can easily be amended by adding a **[cy]** tag, but the point is that the impact of this type of rule can be subtle.

5 Spin-off benefits

The Autoglosser was primarily intended to provide reasonably accurate glosses automatically, thus saving researcher time, but it has also had a number of spin-off benefits which contribute to easier handling of the corpora.

Perhaps the most useful is the ability to use the file contents in the database tables to print out a typeset copy of the transcription in LaTeX, using John Frampton's *ExPex* package.[26] Figure 13 shows the results of this process on the file excerpt previously shown in Figure 1. This greatly facilitates checking for errors in the glossing.

Being able to access the file contents via database queries adds another tool for correcting typos. Selecting all unglossed words in the words

(41) **SER:** **dw** **i** **yn** **hopeless**^{E} **efo** **tynnu** **llun** .
%aut *be.V.1S.PRES.SPOKEN* *I.PRON.1S* *stative.STAT* *hopeless.ADJ* *with.PREP* *take.V.INFIN* *picture.N.M.SG*

I'm hopeless at drawing

(42) **SER:** **dw** **i** **yn** **tynnu** **llun** **i** **yr** **plant** **i** **plant** **i** **yr** **i** **er**$^{C}_{E}$ **Helen**$^{C}_{E}$ **a** **Susanna**$^{C}_{E}$ **a** .
%aut *be.V.1S.PRES.SPOKEN* *I.PRON.1S* *stative.STAT* *take.V.INFIN* *picture.N.M.SG* *to.PREP* *to.PREP* *the.DET.DEF* *children.N.M.PL* *to.PREP* *children.N.M.PL* *to.PREP* *the.DET.DEF* *to.PREP* *er.IM* *name* *and.CONJ* *name* *and.CONJ*

I draw a picture for...for the children, for, er Helen and Susanna and...

Figure 13 – The text in Figure 1 typeset to show alignment of the surface words and their POS-tags.

table gives a list of words which are either unknown because they are not in the dictionaries, or could not be found in the dictionaries because they were mis-spelt (i.e. typos). It is interesting to note that even after two rounds of detailed manual proofreading such typos account for about 0.5% of the words in a file on average, and this technique provides a method of eliminating them.

Autoglossing enforces consistency across the corpus (so that, for instance, Welsh **ychydig** does not appear in some places as *a bit*, and in other places as *a little*), and makes it much easier to change or enrich tags globally. This sort of consistency facilitates data-mining, in that queries can be correspondingly simpler.

6 Further work

Although the current configuration of CG rules is working well, we hope to explore further refinement of the grammar. This would include not only conflating similar rules within a language, but also seeking to use the grammar to mark clause relationships. The latter would be of value in the further linguistic analysis of the influence of clause structure on language switching in bilingual discourse.

Acknowledgments

The support of the Arts and Humanities Research Council (AHRC), the Economic and Social Research Council (ESRC), the Higher Education Funding Council for Wales and the Welsh Assembly Government is gratefully acknowledged. The work presented in this paper was part of the programme of the ESRC Centre for Research on Bilingualism in Theory and Practice at Bangor University.

[26] http://www.math.neu.edu/ling/tex/

References

Lene Antonsen, Trond Trosterud, and Linda Wiechetek. 2010. Reusing grammatical resources for new languages. In *Proceedings of the Seventh Conference on International Language Resources and Evaluation (LREC-10)*. European Language Resources Association.

Bernard Comrie, Martin Haspelmath, and Balthasar Bickel. 2008. Leipzig glossing rules: Conventions for interlinear morpheme-by-morpheme glosses. *http://eva.mpg.de/lingua/resources/glossing-rules.php*.

Kevin Donnelly, Sarah Cooper, and Margaret Deuchar. 2011. Glossing chat files using the Bangor Autoglosser. Paper presented at ISB8, Oslo, May 2011.

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

Brian MacWhinney. 2000. *The CHILDES Project: Tools for Analyzing Talk*. Lawrence Erlbaum Associates.

Jonathan Stammers. 2010. *The Integration of English-origin Verbs into Welsh: A Contribution to the Debate over Distinguishing between Code-switching and Lexical Borrowing*. Verlag Dr. Műller.

Oliver Streiter, Kevin P. Scannell, and M Stuflesser. 2006. Implementing NLP projects for non-central languages: instructions for funding bodies, strategies for developers. *Machine Translation*, 20(4).

OBT+Stat: Evaluation of a combined CG and statistical tagger

Janne Bondi Johannessen, Kristin Hagen and Anders Nøklestad
Text Lab, ILN,
University of Oslo, Norway
{jannebj, kristiha, noklestad} @iln.uio.no

André Lynum
Text Lab, ILN, University of Oslo and IDI, Norwegian University of Science and Technology – NTNU, Norway
andrely@idi.ntnu.no

Abstract

We have created a statistical POS tagger from existing development corpora and use it as a postprocessor to fully disambiguate the detailed morphological and lexical output of a Constraint Grammar tagger. In this article we discuss some of the challenges in unifying these two data-driven and knowledge-based approaches along with the possibilities and challenges that present themselves when using data-driven techniques to disambiguate candidates from a rule-based system. We then present an empirical evaluation that shows how the statistical disambiguation component improves the performance of the rule-based tagger. Our analysis of the results shows the potential for correcting the remaining errors and how the two tagger components interact in the disambiguation task.

1 Introduction

Compared to statistical methods, rule-based systems for natural language processing (NLP) often have a detailed focus on lingustic analysis, and naturally this is reflected in the output from such systems. The constraints and reasoning embedded in the system are often evident in the detail of the system output and ambiguities it leaves unresolved. The Oslo Bergen Tagger (OBT), originally developed at the University of Oslo and University of Bergen[1], is such a system, where linguistic detail has motivated both the particular form of the output and the decision not to fully disambiguate all readings for each token. The precision of the grammar rules and the richness of the underlying lexicon have been both a great resource and a challenge in the later stages of development of the OBT.

1.1 The original OBT output

The OBT is a rule-based tagger based on the Constraint Grammar (CG) formalism (Karlsson et al. 1995), which from the beginning has focused on linguistically precise descriptions. This focus is reflected in the more than 358 morphological tags (with a further 2000 or so for full morphosyntactic analysis) used by the system. Originally the primary users of the tagger were linguists, who are often studying subtle or marginal phenomena, and the recall of the system was considered particularly important; such users want to find all possibly relevant items in a corpus. The CG rules in essence disambiguate between the set of possible readings for a token and have been carefully constructed so that they do not compromise these overall goals of detail and comprehensiveness. As such it was in very few cases seen as acceptable to fuse different tags into portmanteau tags. One illustrative example is the rather large class of nouns that can be treated as either masculine or feminine. We considered it important to be able to identify these as feminine in the few contexts where they are definitely feminine even if that meant the consequence was gender ambiguity in other contexts.

[1] The Oslo-Bergen Tagger was originally developed by the Tagger Project at the University of Oslo in 1996 and written in a rule-based Constraint Grammar (CG1) framework with an interpreter from Lingsoft AB. An electronic lexicon (now available as *Norsk ordbank*) and a preprocessor were also part of the development performed by this project. In 2000 the commercial rule interpreter and the original preprocessor were replaced by components written in Allegro Common Lisp, developed in Bergen at Aksis (now Uni Digital). At this point the tagger was named The Oslo-Bergen Tagger (OBT). In 2008 the OBT was ported to the VISLCG3 framework in order to use the more recent CG3 formalism developed at The University of Southern Denmark in Odense. Uni Digital made a new stand-alone preprocessor based on *Norsk ordbank*, and The Text Laboratory converted the linguistic rules to the CG3 format.

A concrete example that illustrates this and also shows how the OBT works, is the sentence given in (1)[4].

(1) Ei jente drakk. ('A girl drank.')

This sentence is initially rendered by the OBT as in (2), listing all possible readings by lexical analysis. All readings have been supplied by a preprocessor using a rich lexicon, including those that are ambiguous. Note the two possible genders for *jente* 'girl'.

```
   (2)
"<ei/a>"
  "ei"    adv
  "ei"    pron pers sg hum
  "eie"   verb imp
  "en"    det quant fem sing <Correct!>
"<jente/girl>"
  "jente" noun fem com sg indef <Correct!>
  "jente" noun masc com sg indef
"<drakk/drank>"
  "drikke" verb past tense <Correct!>
```

The Constraint Grammar module (Hagen et al. 2000) uses the VISL CG3 rule interpreter to remove all ambiguity that is detectable from the grammatical context, in (2) taking advantage of the preceding feminine determiner. In this example, only those readings marked here with *<Correct!>* are left, as shown in (3).

```
   (3)
"<ei/a>"
  "en"    det quant fem sg <Correct!>
"<jente/girl>"
  "jente" noun fem com sg indef <Correct!>
"<drakk/drank>"
  "drikke" verb past tense <Correct!>
```

The gender ambiguity for *jente* appears in the sentence *Det er bra å være jente.* ('It is good to be (a) girl.'), rendered by the OBT as in (4).

```
   (4)
"<det/it>"
  "det" pron pers 3 sg neut <Correct!>
"<er/is>"
  "være" verb present <Correct!>
"<bra/good>"
  "bra" adj pos neut indef sg <Correct!>
"<å/to>"
  "å" inf marker <Correct!>
"<være/be>"
  "være" verb inf <Correct!>
"<jente/girl>"
  "jente" noun fem com sg indef  <Correct!>
  "jente" noun masc com sg indef <Correct!>
```

We see that *jente* 'girl' became unambiguous in (3), given agreement rules that refer to the feminine determiner, while (4) remains ambiguous since there is nothing in the context to determine the gender of the word. This result was considered acceptable at the time when the OBT was developed, since it would mean that the interested linguist could search a corpus for all instances of feminine nouns, and find ambiguous examples in addition to those where the gender had been fully determined. There are many ambiguities similar to this. Another example is found in the large group of neuter singular indefinite and neuter plural indefinite nouns, which can be disambiguated in some contexts but not all. Also in this case the remaining ambiguity is left in the result on purpose.

OBT only concentrates on grammatical ambiguity and does not look at ambiguity between lemmas. The result is that OBT leaves lemma ambiguities like *fare* 'danger' or *far* 'father' for the ambiguous word form *faren.*

The ambiguity of the OBT is not only of the type we have just described, it also includes a number of unfortunate ambiguities where the CG rules in the OBT do not have enough coverage. In Section 6, we discuss the different kinds of ambiguities more thoroughly.

1.2 Towards unambiguous output in OBT+Stat

While leaving ambiguity in the output can be reasonable from a linguistic standpoint, using the ambiguous output of the tagger as input for other research or engineering purposes constitutes a problem. These often require the output to contain a single reading for each token. Most notably concerning the OBT, the construction of large-scale Norwegian corpora such as *Norsk aviskorpus*[5] and NoWaC (Guevara 2010) requires a high quality, fully disambiguating tagger. Such requirements motivated us to look into how we could make the OBT suitable for this use.

Since continued rule writing activity gave diminishing returns and resources were limited, the need for a statistical module to complement the OBT increased. This motivated the implementation of OBT+Stat, a statistical module that removes all remaining ambiguity from the OBT output, both the grammatical ambiguities and the lemma ambiguities originally left on purpose – and the unfortunate ambiguities.

In this article, we discuss the changes made to the OBT in order to create an effective system,

[4] Tag abbreviations have been translated from Norwegian to English in all examples.

[5] http://avis.uib.no/

OBT+Stat, for fully disambiguated output that still maintains linguistic detail and comprehensiveness.

2 Related work

The combination of CG rule sets with statistical methods was attempted even during earlier work with CG, notably in the combination of the early Xerox XT statistical tagger with the EngCG rules (Tapanainen et al. 1994). This work was reported as showing some promise but does not appear to have been followed up.

Later work described in Hajič et al. (2001) and Spoustová et al. (2007) combines a CG tagger for Czech with statistical models in a sophisticated manner, including decoding that was constrained by the CG tagger output. Their work goes further than the work presented in this article, but since the Czech CG rule set focuses heavily on full recall at the expense of precision, our work is different in scope. Our results are interesting to other CG based systems that aim for high precision at the cost of some recall.

3 OBT+Stat

Since the OBT reaches a high f-score[6] (97.2), it is desirable to keep all complete disambiguations made by the OBT and only add the statistical disambiguation module as a post-processing step in those cases where the OBT leaves some ambiguity. We chose to run a statistical tagger independently of the OBT in a manner that will be described later in this section, and to combine the results instead of attempting more sophisticated modeling based on selecting candidates directly. This results in a simple pipeline where the statistical model is independent of changes in the CG rule set or the lexical preprocessor.

Since there are only a limited number of annotated corpora available for the development of Norwegian NLP tools, we decided to use the corpora already collected and annotated during the development of the OBT. This consists of 122 523 words in 8178 sentences in a hand-annotated development corpus and a corresponding corpus reserved for evaluation with 32 677 words and 2213 sentences. These corpora contained a number of tokens where several readings were marked as correct. Those parts of the corpora had to be annotated again in a fully disambiguated manner[7] - either by combining several tags into one or by keeping the original set of tags and making specific decisions about which reading is considered the correct one. Since combining tags for all readings would erase linguistic detail, and adding such combined tags for the ambiguous tokens would only add considerable complexity to the already large tag set, we chose instead to establish a set of annotation guidelines for the ambiguous cases.

These guidelines by necessity have to reflect some arbitrary decisions. For example, for words like *jente* 'girl', which we recall from Section 1 can be either masculine or feminine, we always choose the feminine tag in ambiguous contexts. In other cases, like words that are ambiguous between singular and plural, we let the human annotator take into account factors like semantic interpretation, knowledge of mass and count distinctions etc. to decide which reading to choose. We discuss the consequences of these guidelines further in Section 5.

As noted, the statistical disambiguation module works by running a statistical tagger independently of the CG-based disambiguation rules. The two results are then unified when the CG rule set leaves more than one reading for a token. If the reading produced by the statistical tagger agrees with one of the readings left by the CG tagger, that reading is selected. If the two taggers do not agree, we attempt to disambiguate using the lemma if possible (as explained at the end of this section), or, failing that, select an arbitrary reading. As we will see in Section 4, the statistical tagger we use covers nearly 80% of the ambiguous readings, with the lemma disambiguation covering most of the remaining ones, leaving only 0.63% to arbitrary selection.

Hidden Markov Model (HMM) based taggers are well known and widely covered in the literature (see e.g. Brants 2000). The HMM model conditions its sequence labelling decisions on the previous one or two labels and the current token to emit. HMM models also include mature and empirically founded models for unknown words when used with most European languages. In our experience, the available HMM taggers, while currently outperformed by more sophisticated models, still provide robust and competitive results on real world data. We have chosen to use the HMM tagger HunPos (Halácsy 2007), which gave good performance on the disambiguation

[6] We use a standard balanced f-score, which is defined as 2 * precision * recall / (precision + recall).

[7] All these corpora were annotated by Arne Martinus Lindstad.

task in addition to being robust and open software with very fast training and decoding performance.

One may ask if the simplistic manner in which we use the statistical tagger is the right way to model the statistical disambiguation process, but it is our intuition that with the scarce resources available more advanced and specialized models, such as constrained decoding or direct discriminative modeling on the disambiguation task itself, may not necessarily yield consistent improvements. The off-the-shelf HMM models offer robust handling of unknown words and well-understood hidden sequence labeling which we regard as more cost-effective in terms of effort and results than a model specifically designed for the task.

Our lemma disambiguation scheme is also simple. It uses the recently created NoWaC corpus of Norwegian documents published on the internet. Our idea is that most lemmas will appear as words in a large corpus since Norwegian lemmas correspond to uninflected words forms. Motivated by this we use a word frequency list derived from NoWaC and select the lemma with the highest frequency in the corpus. This part of the statistical disambiguator considers the output of the CG tagger rather than being run independendently like the POS tagger. We will discuss the lemma disambiguation further in Section 6.2.

4 Comparison with the OBT

Comparing the earlier published evaluation results for the OBT with the fully disambiguated results presents some difficulties since the tasks are different in some aspects and are usually evaluated differently in the relevant literature. The OBT tagger, like other CG based taggers, has previously focused on removing readings that it knows to be incorrect according to the linguistic knowledge embedded in the rules, while leaving the remaining readings in the result. As we have shown in earlier sections, several correct or incorrect tags may be left after disambiguation, and both the precision and recall are reported in evaluations, combined into a standard balanced f-score. CG taggers often make a trade-off between precision and recall where aggressively eliminating readings will increase precision after leaving fewer incorrect readings while reducing recall by unintentionally removing correct readings. Keeping lost recall to a minimum had a high priority during the development of the OBT since it was deemed important that relevant linguistic examples should not be lost when searching a corpus. The OBT for Norwegian *Bokmål*[8] achieved a standard f-score of 97.2, with a recall of 99.0% and a precision of 95.4%, a highly competitive result, but including some ambiguities as we explained in Section 1.

In contrast, when we perform fully disambiguated tagging, the notion of several correct readings disappears, and precision and recall become identical. Results from this kind of tagging are therefore usually reported as a single token wise accuracy score. This accuracy score is not directly comparable to the older precision score since the evaluation corpus is now fully disambiguated and several readings that were previously considered correct are now considered incorrect. One could view the fully disambiguated evaluation as having 100% recall, but the f-score is not a linear function and we can only speculate on the precision of the OBT tagger if rules were developed to raise the recall to this level.

Still we maintain that the two scores can be compared within reason. We will mainly compare the earlier precision score with the accuracy of the new results, considering a slightly lower accuracy for the fully disambiguated result to be at the same level as the older results, and comparable or higher accuracy as an improved result.

5 Evaluation

The training and test corpora for the OBT are drawn from a variety of text types: newspapers (with headlines, by-lines etc.), journal articles, magazines, government reports, and fiction. Combined with the focus on detailed linguistic representation this makes both the underlying data and the resulting analysis more diverse than what is usually the case (much language technology development has been done on for example the homogenous Wall Street Journal corpus). Having presented this reservation, we will first show some statistics measuring the amount of ambiguity in the corpus before presenting the results proper.

After the CG-driven disambiguation on the fully disambiguated test corpus described in Section 3, the amount of ambiguities is as summarized in Table **1**. The still ambiguous tokens now constitute 8.6% of the test corpus, and as seen from the table and the previous discussion, they

[8] The tagger for the Norwegian language variety *Nynorsk* is not discussed in this paper.

result mainly from an inability to choose between two readings.

	Total readings/ tokens	Ratio
Ambiguity over all words	35639 / 32677	1.09
Ambiguity over ambiguous words only	5778 /2816	2.05

Table 1 Amount of ambiguity left by the OBT in the evaluation corpus.

The overall accuracy of the OBT+Stat tagger is measured at 96.56%. We consider this to be a good result considering the fact that we have removed all remaining ambiguity while at the same time kept a large, detailed tag set and disambiguated lemmas with identical tags.

Breaking down the results further, we see from Table 2 that the POS/morphological disambiguation accuracy is slightly higher than the overall accuracy, which includes disambiguation of identical tags with different lemmas in addition to the pure POS/morphological disambiguation.

	Correct readings / tokens	Ratio
Overall accuracy	31552 / 32677	96.56%
Tagging accuracy	31614 / 32677	96.74%
Lemmatization accuracy	32131 / 32677	98.33%

Table 2 Accuracy scores measuring the performance of the fully disambiguating tagger. Scores are shown for tagging and lemmatization separately and combined.

As shown in Table 3, the overlap between the statistical tagger and the OBT for the tokens left ambiguous by the OBT is quite good: nearly 80% of the cases in question. For these words the accuracy of the statistical tagger is 81.70%. Since these are the most difficult tagging decisions left over by the OBT we find this performance of the statistical tagger to be quite good. Errors include some tokens out of coverage where OBT has mistakenly eliminated the correct reading, making it impossible for the OBT+Stat system to make the right decision. The corresponding statistics for the lemma disambiguation are harder to analyze since this model is only used in very specific circumstances, but the coverage and precision indicate that the lemma model is effective and has some impact on the overall result.

	Ratio
Statistical tagger coverage	79.39% (2273/2863)
Statistical tagger accuracy	81.70% (1857/2273)
Lemma model coverage	54.23% (551/1016)
Lemma model accuracy	86.71%

Table 3 The coverage and accuracy of the statistical disambiguation module on the ambiguous tokens for the statistical tagger and lemma disambiguator respectively.

In addition to evaluating the OBT tagger as it is currently in use we also constructed a CG rule set where we removed rules that had been written in order to remove spurious ambiguity by means of heuristics. The premise is that the ambiguities covered by those rules should now be covered by the statistical module. The results of removing these rules are shown in Table **4**.

	correct readings / tokens	Ratio
Overall accuracy	31332/32677	95.88%
Tagging accuracy	31459/32677	96.27%
Lemmatization accuracy	32187/32677	98.50%

Table 4 Accuracy scores for the fully disambiguating tagger with a modified CG rule set where heuristically disambiguating rules are removed.

The overall accuracy using this rule set is slightly lower, 0.68% in absolute and about 20% in relative terms. The change in tagging accuracy is about the same, while the lemma accuracy is a bit higher. This shows that the statistical model does not necessarily handle some of the ambiguities covered by the heuristic rules as well as the rules themselves do. Including the heuristic rules is still beneficial when using the statistical disambiguator.

We also evaluated the statistical module with a CG rule set that attempts to fix some of the disambiguation decisions that have now been annotated in a consistent manner in the training and development corpora based on the new annotation guidelines. Those rules should hopefully consistently determine some ambiguities which the statistical disambiguation module may not resolve in consistent manner. The rules mostly

concern ambiguous masculine/feminine nouns, which are disambiguated as feminine in contexts that lack gender agreement (as illustrated in example (2) in the introduction). The results are shown in Table **5**.

	Correct readings / tokens	Ratio
Overall accuracy	31668/32677	96.52%
Tagging accuracy	31668/32677	96.91%
Lemmatization accuracy	32181/32677	98.48%

Table 5 Accuracy scores for the fully disambiguating tagger with a modified CG rule set that attempts to fix disambiguities now resolved in the annotation guidelines.

The results using these rules are roughly the same as the results for the main CG rule set, indicating that the CG rules and statistical module perform the disambiguation of those cases about equally effectively.

6 Discussion of the results

In this section we will look at some successful and some unsuccessful results, both with respect to grammatical tags and with respect to lemma disambiguation.

6.1 Grammatical disambiguation

We will have a look at the contribution of the statistical module in isolation. We first examine some successful examples.

```
(5)
Offentlige etater har ansvar for…
Public institutions have responsibility for…

Resolved ambiguity:
"<ansvar/responsibility>"
    "ansvar" noun neut com sg indef
             <Correct!> <SELECTED>
    "ansvar" noun neut com pl indef
```

In (5), the OBT had left the word *ansvar* 'responsibility' ambiguous between singular and plural (recall the discussion in Section 1). This particular kind of ambiguity accounts for 615 of the remaining ambiguities before statistical disambiguation, or 21.8%. The statistical module has correctly identified this word as singular, and over all such ambiguities, 418, or 68.0%, are disambiguated correctly by the statistical module.

In example (6) there are actually two ambiguities, both between parts of speech, which have been resolved: one between adverb and preposition reading, and one between verb and noun. The adverb/preposition ambiguity is fairly marginal with only two occurrences in the test corpus, both disambiguated correctly and identical to the one shown in the example. The noun/verb ambiguity is of a more interesting type with 58 (2.1%) occurrences in the corpus. It is often fairly complex with over half of the occurrences having three or more readings to disambiguate and six occurrences having five or more readings. Still, the accuracy of the statistical module for this kind of ambiguity is good, 82.8% (48 correctly resolved occurrences).

```
(6)
... at for mange typer informasjon vil de
elektroniske mediene etter hvert bli enerå-
dende.
... that for many types (of) information the
electronic media will slowly become dominant.

Resolved ambiguity I:
"<for>"
    "for" adv
    "for" prep <Correct!> <SELECTED>

Resolved ambiguity II:
"<typer>"
    "type" noun masc com pl indef <Correct!>
           <SELECTED>
    "type" verb present
```

We now turn to the less successful choices made by the statistical module. In the first quarter of the test corpus, OBT+Stat makes a total of 105 errors. The distribution of those errors with respect to grammatical categories is summarized in **Table 6**.

Singular or plural in neuter nouns (i)	41
Other singular or plural errors (ii)	5
Gender agreement (iii)	15
Nouns ending in -s, genitive or not genitive (iv)	4
Adjective gender (m,f,sg,pl) chosen instead of adj neut/adv (v)	13
Lack of imperative (vi)	3
Other errors (vii)	24

Table 6 Counts of disambiguation errors caused by the statistical module.

Not surprisingly, the largest group of errors by far is due to the difficulty of assigning singular or plural readings to neuter nouns that have the same form in both indefinite singular and plural. As discussed in Section 3, the decisions for the training and test corpus were left to the human annotator, based on linguistic knowledge. The patterns may not in all cases be clear enough for the statistical module to make correct decisions. One may ask whether keeping this distinction as

separate tags apart is a good choice, or whether they should have been collapsed.

It turns out that when classifying errors by grammatical type, they usually have very little in common within the classified category. We will now show a few selected examples, beginning with type (i) in (7):

```
(7)
Generelt om informasjon og særtrekk ved of-
fentlig informasjon
Generally about information and characteris-
tics in public information

Wrongly resolved ambiguity:
"<særtrekk/characteristic>"
       "særtrekk" noun neut com sg indef
                  <SELECTED> <ERROR>
       "særtrekk" noun neut com pl indef
<Correct!>
```

In this example we see that the word *særtrekk* 'characteristic' has been tagged as a singular noun, while the test corpus assigns it a plural interpretation. There is very little in the grammatical surroundings that would give a hint as to the correct interpretation. The most likely (but incorrect) clue would have been the fact that this word is a conjunct in a coordination phrase, and that the first conjunct is a singular noun. However, in the present case, the first conjunct is a mass noun, while the second conjunct is a count noun, and it is only when this is taken into account that the right plural tag can be applied. The mass/count distinction is not marked in the tags. A larger training corpus could possibly allow OBT+Stat to disambiguate this case properly.

Our example (8) is of type (ii), but still one that deals with singular and plural.

```
(8)
Selv om utvalget kanskje særlig viser noen av
Hjemme-PCs favoritter
Even if the selection perhaps especially show
some of Hjemme-PC's favourites

Wrongly resolved ambiguity:
"<noen/some>"
       "noen" pron pers 3 sg masc fem
              <SELECTED> <ERROR>
       "noen" pron pers 3 pl <Correct!>
```

In (8), the error is in the assignment for number for the word *noen* 'some/any'. Like with the wrongly annotated word in (7), it is not in an agreement context, which is most likely the reason that the OBT did not disambiguate it. The interpretation of *noen* as singular is actually only possible when it is used as a negative polarity item, which is obviously not the case here. However, since there are a variety of constructions that license negative polarity items in Norwegian (Lindstad 1999), the CG tagger was not able to make the correct choice.

A third type shown in example (9), type (vi), illustrates the problem of having short headlines in the test corpus.

```
(9)
Fly
Fly

Wrongly resolved ambiguity:
"<fly>"
"fly" noun neut com sg indef
      <SELECTED> <ERROR>
"fly" noun neut com pl indef
"fly" verb imp  <Correct!>
```

Even for a human annotator it may be difficult to interpret what the headline is actually meant to be; imperative or noun. Imperatives happen to be few in written texts, and as a result, a statistical module will almost invariably fail in this task.

At the end of this section we would like to point out that while it could have been conceivable that the statistical module made many of its errors due to already faulty output from the OBT (i.e., with the correct tag missing), a quick count shows this not to be the case. Out of our 749 errors made by the statistical module, a modest 74 are due to errors made by the OBT tagger.

6.2 Lemma disambiguation

The OBT was never developed with the intention of doing lemma disambiguation. Often ambiguous lemmas have different grammatical characteristics and they would effectively be disambiguated anyway, but this is not always the case. There were 515 ambiguous lexical lemmas, out of which 395 (76.70%), were correctly resolved. Looking at the 28 errors in the first quarter of the test corpus we will first give some examples of successfully resolved lemmas, and then some less fortunate ones.

```
(10)
I alle deler av den offentlige forvaltningen
In all parts of the public administration

Resolved lemma ambiguity:
"<deler>"
    "del/part" noun masc com pl indef
               <Correct!> <SELECTED>
    "dele/border" noun neut com pl indef
```

In (10), the ambiguity is between the lemmas *del* 'part' and *dele* 'border', and it is only the first one that is correct, correctly disambiguated by OBT+Stat. Given that the correct word has a much higher frequency than the other, we would expect that the tagger chose correctly.

```
(11)
En positiv utvikling i statlig informasjons-
virksomhet de siste årene
A positive development in state information
practice the last years

Resolved lemma ambiguity:
"<årene>"
    "år/paddling ore" noun fem com pl def
    "år/paddling ore" noun masc com pl def
    "år/year" noun neut com pl def
              <Correct!> <SELECTED>
    "åre/paddling ore, vein" noun fem com pl
                             def
    "åre/paddling ore, vein" noun masc com pl
                             def
```

In (11), with all the lemmas to choose between, it is satisfying that OBT+Stat made the right choice. Given that the right word is the most general one, and hence occurs in a wide variety of texts, this is the most frequent word form found in NoWaC and subsequently chosen by the tagger.

However, the texts in the test corpus do not always deal with the most general topics. Hence our first illustration of an error shows the same word form, but now with a different lemma as the correct one. Consider (12).

```
(12)
Fysikeren ville f.eks. studere sammenhenger
mellom blodtrykk, størrelse av årene og blod-
tilførsel.
The physicist wanted e.g. (to) study connec-
tions between blood pressure, size of the
veins and blood supply.

Wrongly resolved lemma:
"<årene>"
"år/paddling ore" noun fem com pl def
"år/paddling ore" noun masc com pl def
"år/year" noun neut com pl def
          <SELECTED> <ERROR>
"åre/paddling ore, vein" noun fem com pl def
                         <Correct!>
"åre/paddling ore, vein" noun masc com pl def
```

The text is about a medical topic, and here the less general lemma meaning 'vein' is the correct one, which the statistical module did not find. We have looked at the test corpus, and the word form *årene* occurs six times, five of them in the 'year' meaning. Sometimes, however, the frequency does not seem to be so equally distributed in NoWaC and the test corpus. Consider (13).

```
(13)
Faren min var mye ute og reiste
My father was much out and travelled
(=travelled a lot)

Wrongly resolved:
"<faren>"
    "far" noun masc com sg def <Correct!>
    "fare" noun masc com sg def
           <SELECTED> <ERROR>
```

The statistical module should disambiguate between the lemma meaning 'father' ("far") and that meaning 'danger' ("fare"). One would guess that in both corpora the word meaning 'danger' would be more frequent, and in fact in (13) the statistical module has picked 'danger' due to an evidently higher frequency of the word form *fare* than of *far* in NoWaC. For this example the NoWaC and OBT corpus disagree in the distribution of the semantics of this word form. In fact, out of four occurrences of the word form *faren* in the test corpus, three had the meaning 'father' and only one the meaning 'danger'.

Our conclusion with respect to the lemma module is that it seems to work quite well, since most lemmas have large differences in frequency as word forms, and the differences seem to correspond fairly well between NoWaC and the test corpus. Furthermore, the hypothesis that uninflected word form frequencies in a large corpus can be used as an indication of lemma frequencies seems to bear out in practice. It is still possible that some rule-based approach would improve our lemma disambiguation. For example, the word form meaning 'father' very often occurs together with a possessive pronoun, and this kind of knowledge could have been put into the system.

7 Conclusion

We have improved an originally rule-based tagger, the OBT, with a statistical HMM tagger. The latter has been applied on the still ambiguous output of the OBT. The resulting OBT+Stat system performs well and has two added advantages compared with the original tagger in that it gives unambiguous output and it performs lemma disambiguation.

References

Alfred. V. Aho and Jeffrey D. Ullman. 1972. The Theory of Parsing, Translation and Compiling, volume 1. Prentice-Hall, Englewood Cliffs, NJ.

American Psychological Association. 1983. Publications Manual. American Psychological Association, Washington, DC.

Association for Computing Machinery. 1983. Computing Reviews, 24(11):503-512.

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. Journal of the As-

sociation for Computing Machinery, 28(1):114-133.

Brants, T. 2000. TnT: a statistical part-of-speech tagger. In Proceedings of the Sixth Conference on Applied Natural Language Processing. Seattle, Washington.

Emiliano Guevara. 2010. NoWaC: a large web-based corpus for Norwegian. In Proceedings of the NAACL HLT 2010 Sixth Web as Corpus Workshop. Los Angeles.

Giesbrecht, Eugenie and Stefan Evert. 2009. Is Part-of-Speech Tagging a Solved Task? An Evaluation of POS Taggers for the German Web as Corpus. In Proceedings of the Fifth Web as Corpus Workshop (WAC5). San Sebastian, Spain.

Hagen, Kristin, Janne Bondi Johannessen and Anders Nøklestad. 2000. A Constraint-based Tagger for Norwegian. In 17th Scandinavian Conference of Linguistics, [Odense Working Papers in Language and Communication 19], Carl-Erik Lindberg and Steffen Nordahl Lund (eds), pp. 31-48. University of Southern Denmark, Odense.

Jan Hajic˘, Pavel Krbec, Pavel Kve˘ton˘, Karel Oliva and Vladim´ır Petkevic˘. 2001. Serial Combination of Rules and Statistics: A Case Study in Czech Tagging. In Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics. CNRS – Institut de Recherche en Informatique de Toulouse and Universite´ des Sciences Sociales, pp. 260–267. Toulouse.

Péter Halácsy, András Kornai, Csaba Oravecz. 2007. HunPos - an open source trigram tagger. In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics. Companion Volume Proceedings of the Demo and Poster Sessions. Association for Computational Linguistics, Prague, Czech Republic.

Johannessen, Janne Bondi and Kristin Hagen. 2003. Parsing Nordic Languages (PaNoLa) norsk versjon. In Nordisk Sprogteknologi 2002, pp. 89-95. Museum Tusculanums Forlag, University of Oslo.

Karlsson, Fred, Atro Voutilainen, Juho Heikkilä and Arto Anttila. 1995. Constraint Grammar, A language independent system for parsing unrestricted text. Mouton de Gruyter. Berlin; New York.

Lindstad, Arne Martinus. 1999. Issues in the Syntax of Negation and Polarity in Norwegian. A Minimalist Analysis. Cand.philol thesis, University of Oslo.

Spoustová, D., J. Hajič, J. Votrubec, P. Krbec and P. Květoň. 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. Prague, Czech Republic.

Tapanainen, P. and A. Voutilainen. 1994. Tagging accurately: don't guess if you know. In Proceedings of the Fourth Conference on Applied Natural Language Processing. Stuttgart, Germany.

VISL CG3: http://beta.visl.sdu.dk/cg3.html

A Finite State Constraint Grammar Parser

Janne Peltonen
University of Helsinki
Helsinki, Finland
janne.peltonen@helsinki.fi

Abstract

It has long been held that finite state (FST) methods should be the method of choice in implementing a CG parser (e.g. Karlsson (1990)), since FST methods are very well understood mathematically and typically quite efficient. However, more or less all implementations to date have been using other methods. I set out to bridge the gap between the FST world and the CG world[1]. I created a representation for ambiguous, morphologically analysed sentences that might be general enough to be used in other projects as well. I was able to create a compilation procedure for most types of **CG-2** rules into an FST format, and successfully apply the compiled rules to my sentence representation. I implemented the grammar file parsing and rewrite rule generation using **Python 3**, and used Måns Huldén's (2009) **Foma** for the actual FST operations. I also evaluated the implementation in terms of time and space requirements.

1 Previous Work

My research is far from being the first attempt to combine the worlds of Constraint Grammars and finite state methods. For example, the *Finite State Intersection Grammars* (FSIG) developed by Koskenniemi (1990) are a decidedly CG-like finite state approach to disambiguation and surface syntactic parsing — so much so that FSIG has been suggested to be renamed Parallel CG and traditional CG, Sequential CG (Voutilainen, 1994). In FSIG, constraints act on complete sentences and prune complete readings on a sentence level, whereas traditional CG prunes readings from individual cohorts (word forms).

Gross defines *local grammars* loosely as a class of grammars that reduce ambiguity by local constraints (Gross, 1997). Mohri (2005) shows an algorithm to disambiguate an ambiguous sentence automaton using only local constraints. This, too, can be considered as a CG-like FST approach to disambiguation.

Graña *et al.* (2003) show a method to compile constraint-based textual rules directly to FSTs. However, their rule format differs somewhat from the previously used formats. My goal was to be backwards compatible with Tapanainen's **CG-2** (Tapanainen, 1996), since there are lots of grammars written using that formalism. I didn't use the open source **VISL CG-3** formalism since its differences from **CG-2** are minor — and the formalism is currently in a state of rapid evolution. Additionally, the Swahili grammar sample I was allowed to use in my work was written in **CG-2**.

2 Ambiguity Representation

One obvious problem to solve was the representation of sentences as finite state automata. There must be a representation for local ambiguity either (i) as word "lattice" containing paths that represent combinations of local readings; (ii) as a pearl chain shaped compressed word lattice where the dependencies across word boundaries are not maintained, but every combination have a separate path; or as (iii) as a single path automaton containing a string that lists all the local ambiguity classes (aka cohorts) on the single line.

The reasons for choosing the last methods — single path sentence automata — are, in no particular order, as follows:

- Avoiding the possibility of exponential search times.

[1] The results presented herein are also published in my master's thesis in Finnish in June, 2011.

- Apparent straightforwardness of composing a single path automaton to the rule automata.
- Possibility to refer to sibling readings, that is, readings in the same cohort.
- Usability of the representation even if the implementation of rules differs radically from the currently adopted one.

The actual form I chose is as follows:

- Cohorts are separated by the symbol ¤:
  `¤ cohort ¤ cohort ¤ ... ¤ cohort ¤`
- Readings are separated by the symbol §:
  `§ reading § reading § ... § reading §`
- There is a cohort separator at the beginning and end of the sentence, as well as a reading separator before the first reading and after the last one
- The word form is between the cohort separator and the first reading:
  `¤ "<word-form>" § reading §`
  `...`
- The base form is the first tag in the reading:
  `... § "base-form" TAG1 TAG2`
  `...`
- The word form is repeated as the last tag in each reading:
  `... § "base-form" TAG1 TAG2`
  `... "<word-form>" §`

Figure 2 shows the sentence automaton for two word forms (cohorts) in an ambiguous sentence. In the traditional vertical form, the sentence would appear as in example 1.

(1)
```
...
"<sm1>"
        "pm11" P111
        "pm12" P121 P122
"<sm2>
        "pm21" P211
        "pm22" P221
...
```

3 Rule Representation

A natural form to use for the CG disambiguation rules themselves was Lauri Karttunen's Replace rule syntax (Karttunen, 1995) — there was an existing, open implementation available, and the rule formalism appeared strong and simple enough. The only problem with replace rules were **CG-2**'s (Tapanainen, 1996) linked rules: theoretically, a linked rule's left context might span to the right side of the rule target, and there is no easy or easily generalisable way to represent a left context that might leak to the right side of the rule centre in Karttunen's formalism. I chose to treat linked rules as a special case treated later. In the test grammar I have available there were only four linked rules, so that limitation appeared to be within reason.

I wanted the application strategy of the rules to mirror what I understood of current **CG-2** parsers as closely as possible. That is, at the rule level, I wanted the rules to appear to proceed from left to right — the left context of a given rule had to have the appearance of the rule being already applied there whereas the right context should be uncharted territory (Tapanainen, 1996). So the obvious variant of Karttunen's rules was the right-oriented (`//`) one — left context from the lower, or output, tape; right context from the upper, or input, tape.

The choice of right-oriented replace rules created interesting complications in the replace rules, especially when combined with the robustness clause 'the last reading of a cohort shall not be removed'. To elaborate: if a left context is to apply, the readings that contain the left context must not be marked as erased — or, in the case of a negated left context, they must all be marked as erased. But if all the readings in the cohort are marked as erased, then the last reading of the cohort should be treated as non-erased after all, because there has to be at least one reading left in each cohort. To be consistent with the left-to-right application strategy, that should be the rightmost one.

A simple first approach for rule 2 would be as in example 3. Here, a `REMOVEREADING` tag is added next to the target tag if there is a context tag `CTAG` in the previous cohort. `.¤.` means all strings that contain at most one cohort separator.

(2)
```
REMOVE (TTAG) IF (-1 (CTAG));
```

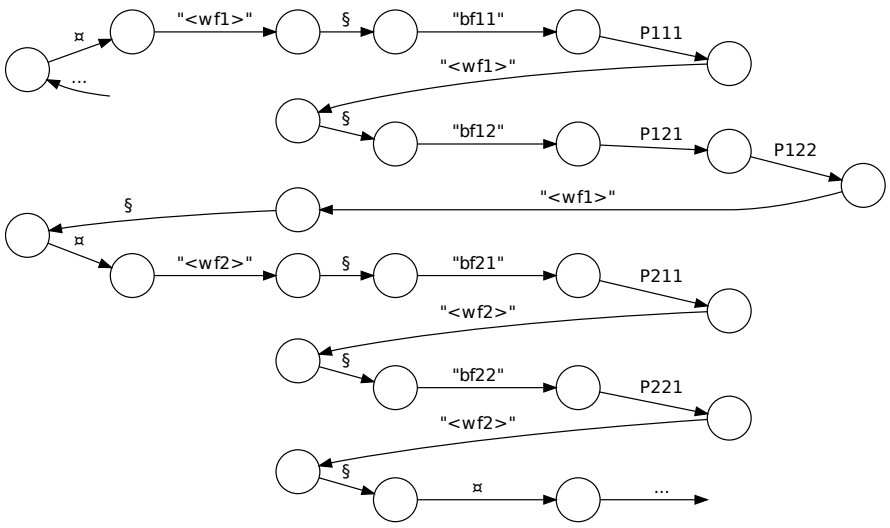


Figure 1: A Single Path Sentence Automaton

(3)
```
TTAG -> TTAG REMOVEREADING ||
  CTAG .¤. _ ;
```

However, rule 3 doesn't reflect the internal rule application strategy outlined. To achieve that, a more complicated approach is needed. The end result is represented as rule 4.

(4)
```
TTAG -> TTAG REMOVEREADING //
  .#. ... [
   ¤ ..
    § .nRR CTAG .nRR §
      .. |
   ¤ .
    [§ . REMOVEREADING .]*
     § . CTAG . §
  ] ¤. _ ;
```

The left context of rule 4 is composed as follows:

- `.#. ...` anchors the context to the beginning of the sentence. This is needed when combining context constraints and wouldn't be strictly necessary in this rule, with only one constraint.
- Two possibilities for the context cohort follow:
  1. The cohort contains a reading with the context tag `CTAG` and no `REMOVEREADING` tags (`.nRR` matches everything in a reading except a `REMOVEREADING` symbol); or
  2. all the previous readings in the cohort are marked as removed, so we don't care if the last reading is marked as removed — if it contains a `CTAG`, it matches.
- `¤.` means one immediate cohort separator and anything inside a cohort after that.

On higher level, that is, the levels of rule sets and grammar files, I chose to mimic the rule application order of Pasi Tapanainen's **CG-2** implementation (Tapanainen, 1996). First, a working set of rules is generated from the first constraint section in the grammar file, in the order they appear in the grammar file[2]. Then, for the sentence being processed, each rule is applied in isolation, and a new sentence automaton is created from the result of rule application — if there was a change; otherwise, the original sentence automaton is used with the next rule. When all the rules in the working set have been applied once, the whole process is repeated, until the working set of rules can no longer change the sentence automaton. At this point, the rules from the next constraint section in the grammar file are appended to the end of the working set, and the process is repeated until there are no more constraint sections left.

Changing the application order of rules would be relatively easy, since the rule application order

[2]This is actually different from Tapanainen's implementation which gives no guarantees about the application order of rules inside a constraint section; in Tapanainen's implementation, only the application order of constraint sections is defined to match the order of the sections in the grammar file.

is mostly defined in **Python**. On the other hand, running multiple rules in parallel for one sentence — in essence, trying to disambiguate each cohort in a sentence with all rules before advancing to the next cohort — is not easily achievable with my approach. It could be doable, in theory, by combining the context constraints of all the rules to create one huge replace rule. However, very complex replace rules appear to be slow to handle.

4 Implementation

The CG rule parser/re-writer was written in **Python 3**[3], using the PLY (**Python Lex-Yacc**) parser generator[4]. Måns Huldén's **Foma**, as a sub-process of the **Python** script, is used to compile the replace rules into transducers. The conversion between the traditional CG sentence format and the new sentence format is done in pure **Python**. **Foma**, as a sub-process of the **Python** script that controls the high level application order of rules, is also used to apply the rules to sentences.

To create the compilation process from CG rules to rewrite rules, I went through the different disambiguation rule types in Pasi Tapanainen's **CG-2** version of the CG formalism and came up with a rewrite rule equivalent for each of them. In the current implementation, the rewrite rule generation phase isn't as elegant as it could be: the rule generator goes through all different combinations of **CG-2** rule features and creates a rule for each combination separately, even if some of the features could be treated as modifiers of the original rule. For example, I suspect that I could implement rule negation as a simple textual transformation of the positive version of the same rule.

Using **Foma** with **Python** proved to be quite simple: the sub-process modules in **Python**'s standard library provided me with sufficient means to create a sub-process for **Foma** and communicate with it. Currently, I only have a simple implementation that reads lines from the **Foma** sub-process until a certain known line is reached, but I've been experimenting with a separate thread for communication, to avoid accidental lockups. So far, the results have been encouraging and simple to implement.

The implementation, in its current form, is not really distributable. I plan to create a distributable package and make it available on-line soon. In the meantime, I can provide the interested with the current version and instructions on how to make it function.

[3] `http://www.python.org/download/releases/3.0/`

[4] `http://www.dabeaz.com/ply/`

5 Test Grammar And Sentence Data

Professor emeritus Arvi Hurskainen allowed me to use a sample of his **SwaCG** Swahili language grammar to test and develop my own disambiguator (Hurskainen, 2004). The sample contains 397 rules of which 380 are select rules and 17 are remove rules; the approach used in this grammar was more to describe sufficient contexts for certain morphological choices than to describe when some readings should be discarded.

Four select rules contained a linked contextual test. My current implementation ignores rules with linked tests, so the results were, in effect, obtained with a grammar of 393 rules of which 376 are select rules and 17 remove rules.

The morphologically analysed test sentences were also provided by professor Hurskainen. They are a set of 684 sentences, hand-crafted to test different aspects of the Swahili grammar. The sentences are categorised as follows:

- constructions with inflecting adjectives (287 sentences);
- constructions with uninflecting adjectives (133 sentences);
- demonstrative before noun, inflecting adjectives (183 sentences); and
- demonstrative before noun, uninflecting adjectives 81 sentences.

The ambiguous morphological analysis contains 5191 word forms and 11 455 readings, with punctuation included. With punctuation excluded, there are 3139 word forms and 9 403 readings.

6 Results

Rules are applied one at a time, so the rules don't have to worry about other rules interfering with their execution. This also applies when creating compositions of rules: conceptually, the upper tape of each new rule is the lower tape of the previous rule composed with previous rules and the sentence automaton, so the effect is the same as with applying the new rule to a new version of the sentence automaton.

6.1 Space Complexity

Memory requirements didn't appear to grow especially fast when composing more and more rules to the composition that begins with the sentence automaton. On the other hand, when trying to compose as few as two big rules into a composition rule without the sentence automaton, I could easily run out of memory on my workstation. For example, composing two rule transducers of sizes 2.7 KB and 21.2 KB — with two simple special transducers that actually erase the readings that are marked as erased, and clean up superfluous erase markers, between the rules — generates a transducer of size 341.8 KB. Composing rules of sizes 21.2 KB and 121.1 KB result in a transducer of size 4.5 MB, and trying to compose two rules of size 26.4 MB results in **Foma** finally running out of memory on my workstation, after having allocated more than 1.2 GB. 26.4 MB rules are rare, but at this growth rate, 26.4 MB rule compositions wouldn't be.

The sizes of the compressed transducer files, in **Foma** binary representation, vary between 852 B and 2.6 MB — there correspond to uncompressed transducers of sizes 668 B and 26.4 MB. The binary representation of the compiled grammar, with 379 compressed binary rules, takes 8.0 MB.

6.2 Time Complexity

Compilation of the test grammar of 393 rules took at most 90 s (plus 25 s for the packing and unpacking of the binary transducers). **VISL CG-3**'s **CG-2** compatibility mode is 1 500 times faster. On the other hand, the time was minutes, not hours, so it's almost usable.

Disambiguating the test set of 684 sentences created almost the same results as disambiguating with **VISL CG-3**. All the differences could be explained by the fact that **VISL CG-3**, as opposed to Tapanainen's **CG-2** parser and my program, does not take the order of tags into account neither in combined tags nor in same position tests with catenation. Moreover, **VISL CG-2** collapses multiple instances of same tag in a catenation into one (Tino Didriksen, personal discussion). This result is encouraging.

Less encouraging is the result that disambiguating the complete test set took 64 min 12 s. Again, **VISL CG-3** was 1 500 times faster. The reason for the apparent slowness of my approach is not completely clear. An obvious first guess would be overhead in inter-process communication or problems with the speed of the finite state tools used. As it turned out, there were a couple of issues. However, the results given in this chapter are obtained after having resolved most of the technical issues.

6.3 Attempts to Increase Speed

It appears that adding a new rule to a composition of the sentence automaton and other rules takes more or less the same time than composing the first rule to the sentence automaton, or perhaps slightly more. So it is possible to reduce the run time of the program by using longer rule compositions on each iteration, since the overhead of reading and writing data between processes decreases (since the number of iterations decreases). However, there appears to be a cutoff point of something like 20 rules per composition after which the decrease in overhead can no longer compete with the increase in composition time — at least with my test grammar and data.

One complication in communicating with **Foma** was that **libreadline**[5] calls sometimes took a really long time to complete — but without **libreadline**, I couldn't get the inter-process communication to work. A one line patch to **Foma**, to flush its standard output after the completion of a command, solved that problem, and the time to read a longish sentence representation into a **Foma** variable dropped from more than 200 ms with **libreadline** to approximately 40 ms without it.

There were a couple of memory leak issues within **Foma** that Måns Huldén was kind to fix more or less immediately. Also, he provided me with optimised versions of his replace rule translation formulae that sped up the grammar compilation considerably.

6.4 Analysis

The slow results given in section 6.2 are obtained after the modifications made in the last section. Thus it appears that at least currently, the tightest bottlenecks are elsewhere than in inter-process communication or tool errors.

According to the analysis I performed using the **Python cProfile** library[6], most of the time was actually spent composing the sentence and rule transducers together. This would indicate that

[5] `http://www.gnu.org/software/readline/`

[6] `http://docs.python.org/release/3.1.3/library/profile.html`

my program creates so complex replace rules that **Foma** can no longer handle them efficiently. As the most complex rule transducers have tens of thousands of states and nearly two million arcs, that is hardly surprising — the worst case has 32 219 states and 1 732 204 arcs. I have yet to find out why some rules are so complex and what, if anything, could be done to avoid such complexity.

7 Conclusion

In this paper, I have shown that it is possible to create a finite state CG implementation that is mostly compatible with **CG-2**. I was also able to create a useful finite state representation for the ambiguous sentences, to which the rules could be readily applied. Lauri Karttunen's replace rules proved to be a usable basis for representing CG rules. My implementation is not complete, and currently only disambiguates a grammar containing at most the rule types in my test grammar — to test the disambiguator with other grammars, the remaining rule types should be catered for.

However, even if I have provided a proof of concept implementation of a CG disambiguator, my program is too slow for any practical purposes. Additional research is called for. The program might be sped up a bit by replacing inter-process communication with direct library calls, once the required **Python** library bindings are in place. It might also be possible to simplify the generated replace rules. If these approaches fail, other finite state methods than replace rules should be considered. As indicated in personal discussion, at least Anssi Yli-Jyrä and Måns Huldén are currently planning such approaches.

References

Jorge Graña, Gloria Andrade, and Jesús Vilares. 2003. Compilation of constraint-based contextual rules for part-of-speech tagging into finite state transducers. In Jean-Marc Champarnaud and Denis Maurel, editors, *Implementation and Application of Automata*, volume 2608 of *Lecture Notes in Computer Science*, pages 128–137. Springer Berlin / Heidelberg.

Maurice Gross. 1997. The construction of local grammars. In Emmanuel Roche and Yves Schabes, editors, *Finite-state language processing*, pages 329–354. MIT, Cambridge (MA), USA.

Mans Hulden. 2009. Foma: a finite-state compiler and library. In *Proceedings of the Demonstrations Session at EACL 2009*, pages 29–32, Athens, Greece, April. Association for Computational Linguistics.

Arvi Hurskainen. 2004. Optimizing disambiguation in swahili. In *Proceedings of Coling 2004*, pages 254–260, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Fred Karlsson. 1990. Constraint grammar as a framework for parsing unrestricted test. In *COLING-90: papers presented to the 13th International Conference on Computational Linguistics: on the occasion of the 25th anniversary of COLING and the 350th anniversary of Helsinki University*, pages 168–173, Helsinki. International Conference on Computational Linguistics.

Lauri Karttunen. 1995. The replace operator. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 16–23, Morristown, NJ, USA. Association for Computational Linguistics.

Kimmo Koskenniemi. 1990. Finite-state parsing and disambiguation. In *COLING-90: papers presented to the 13th International Conference on Computational Linguistics: on the occasion of the 25th anniversary of COLING and the 350th anniversary of Helsinki University*, pages 229–232, Helsinki. International Conference on Computational Linguistics.

Mehryar Mohri. 2005. Local grammar algorithms. In Lauri Carlson, Antti Arppe, Mickael Suominen, Krister Lindén, Jussi Piitulainen, Martti Vainio, Hanna Westerlund, Anssi Yli-Jyrä, Juho Tupakka, and Markus Koljonen, editors, *Inquiries into Words, Constraints and Contexts. Festschrift for Kimmo Koskenniemi on his 60th Birthday*, CSLI Studies in Computational Linguistics, pages 84–94. CSLI Publications, Stanford, CA, USA.

Pasi Tapanainen. 1996. *The constraint grammar parser CG-2*. University of Helsinki, Department of General Linguistics, Helsinki.

Atro Voutilainen. 1994. *Designing a Parsing Grammar*. University of Helsinki, Department of General Linguistics, Helsinki.

FinnTreeBank: Creating a research resource and service for language researchers with Constraint Grammar

Atro Voutilainen
Department of Modern Languages
University of Helsinki
atro.voutilainen@helsinki.fi

Abstract

This paper described ongoing work to develop a large open-source treebank and related Finnish language resources for the R&D community, especially corpus linguistic researchers. Initially, we look at user needs and requirements that these set for corpus annotation. We propose the linguistic Constraint Grammar as a framework to answer the requirements. The second half of the paper describes ongoing work in the FinnTreeBank project to answer these objectives.

1 Needs of corpus linguists

Language researchers need empirical data to help them formulate and test hypotheses e.g. about natural language grammar and meaning. Morphologically annotated (or POS-annotated) text corpora have been available to researchers for many years, and currently such tagged corpora for many languages are accessible. Some of these corpora are very large, even billions of words (e.g. German COSMAS II). Though automatic tagging tends to misanalyse a few words in a hundred, automatically tagged corpora are generally of sufficient quality and quantity for researchers to enable basically word oriented queries and corpus searches in a local context (e.g. "Key Word In Context").

However, corpus linguists are often interested in phenomena that involve more than local character strings: lexically or semantically motivated units in linguistic context (e.g. as part of a syntactic structure). Extraction of such, often non-local, linguistic patterns is difficult with string-based corpus searches: queries on POS-tagged corpora to recover clause or sentence level syntactic constructions result in too low accuracy (combination of precision and recall), and the amount of manual postprocessing needed to make the data usable for further analysis is too high to make such searches productive.

1.1 Requirements for syntactic annotation

A corpus with an additional layer of syntactic annotation (e.g. phrase structure or dependency structure) is needed to enable successful queries for clause or sentence level syntactic constructs. To enable successful extraction of desired lexico-syntactic patterns (multiword units with(in) the desired syntactic structure), the syntactically parsed corpora need to have a high correctness rate: most sentences in the parsed corpus ('treebank') should have a correct lexical and syntactic analysis.

Further, to enable extraction of patterns containing mid- or low-frequency lexical units in sufficiently high volume for meaningful quantitative analysis, the parsed corpus also should be very large, probably of a size comparable to the largest POS-tagged corpora now available to researchers.

1.2 Limitations with current treebanks

Syntactically parsed corpora, generally referred to as treebanks, are now available for a growing number of languages (cf. Wikipedia entry for "Treebank"), with phrase structure annotations, or, increasingly, with dependency syntactic annotation (to enable analysis of unbounded, or long-distance, dependencies). Most syntactically annotated corpora are very limited in size – typically with thousands, or at most tens of thousands, of sentences (cf. e.g. (Mikulova et al., 2006), (Kromann, 2003) and (Haverinen et al., 2009)).

Assuming corpus linguists are interested in phenomena that involve lexical and syntactic information (involving corpus searches with lexical and syntactic search keys or patterns), a corpus with, say, a 50,000 sentences or a million words, will likely provide far too few 'hits' for such complex queries to enable quantitatively meaningful stud-

ies. To enable a coverage comparable to local-context lexically oriented searches on POS-tagged corpora, syntactically annotated corpora should be even larger than comparable POS-tagged corpora.

1.3 Limitations with complete sutomatic parsing

Automatic syntactic annotation could be proposed as the obvious solution for providing very large syntactically annotated corpora for researchers. However, automatic syntactic corpus annotation is generally avoided in treebanking efforts, probably because the error rate of automatic syntactic analysis is prohibitively high: even the best statistical dependency parsers (such as Charniak, 2000) assign a correct dependency relation and function to slightly over 90% of tokens (words and punctuation marks). If every tenth word is misanalysed, most text sentences get an incorrect syntactic analysis.

Instead, syntactic corpus annotation is done manually (with some level of supporting automation). At a recent treebank course (organised by CLARA in Prague, December 2010), some of the presenting treebank projects reported manual annotation times at 5-20 minutes per sentence, and there were reports of nearly decade-long treebanking efforts resulting in treebanks of some tens of thousands of sentences.

In the current language technology community, automatic syntactic modelling and analysis are usually carried out with data-driven language models that are based on statistics generated from manually annotated treebanks. Statistical models based on scant or inconsistent data frequently mispredict; even at the lower levels of linguistic analysis with larger quantities of available training data, POS taggers with statistical language models mispredict the category of several words in a hundred (which means that close to or more than half of all sentences are tagged incorrectly). The best statistical dependency parsers reach labelled attachment scores of slightly above 90% at word level in optimal circumstances (training text genre is the same as that of evaluation corpus); for many other languages, the labelled attachment scores reported are substantially lower. With accuracy scores of this magnitude at word level only, incorrectly parsed sentences are likely to constitute the vast majority of all parsed sentences. In short, current statistical models of syntax are probably too inaccurate to provide a complete solution to high-quality automatic treebanking.

To sum, large-scale treebanking efforts seem to be in a deadlock: manual treebanking is too work-intensive (and possibly also too inconsistent) to enable creation of sufficiently large treebanks to support statistically significant corpus linguistic research; statistical parsing efforts so far have failed to provide sufficiently high parsing accuracy to enable automatic creation of high quality research data for corpus linguists.

2 Constraint Grammar as a solution

Constraint Grammar is a reductionistic linguistic paradigm for tagging and surface-syntactic parsing (Karlsson & al, 1995) that has the following properties to make it an attractive environment for treebanking purposes.

- Large-scale work on tagging and parsing has been done in this framework on several languages since late 1980s (cf. Wikipedia entry on Constraint Grammar)
- The most advanced publicly available implementation of the compiler-interpreter (VISL cg3) supports a wide range of functionality, from lexical analysis to disambiguation to dependency syntax
- A grammarian makes and modifies language models (lexicons, parsing grammars), with very competitive accuracy (measured e.g. as precision-recall tradeoff) and modifiability
- CG tagging and parsing can yield full or partial analysis, which enables a necessary control on precision-recall tradeoff for different purposes such as treebanking

As an example case, we consider an early evaluation and comparison on word-class tagging in English (Samuelsson and Voutilainen 1997). In this report, EngCG-2, the second major version of the English Constraint Grammar for word-class disambiguation, was compared with a state-of-the-art statistical ngram tagger (Hidden Markov Model), to answer certain open questions about the original ENGCG by the research community of the late 1990s.

For the experiments, an common tag set and corpora were documented and used, with options for full and partial disambiguation. In EngCG-2,

the disambiguation grammar was organised into five increasingly heuristic subgrammars to enable trading recall for precision.

Regarding precision-recall tradeoff of the two taggers in the experiment (cf. Table 1 in the Samuelsson and Voutilainen article), the main observations are:

- With almost fully disambiguated outputs, the ngram tagger discarded a correct analysis 9 times more often than EngCG-2.
- When more ambiguity was permitted in the taggers' analyses, the ngram tagger discarded a correct analysis 28 times more often than EngCG-2.

The possibility to make almost safe predictions in a linguistics-based parsing environment, to control the precision-recall tradeoff, and to achieve a very competitive precision-recall tradeoff is shown in this comparison.

Though we are unaware of similar comparisons at the level of dependency syntax (assignment of dependency functions and dependency relations to words), similar control on the tradeoff between accuracy and partiality of dependency syntactic analysis can be exerted in CG: the rule formalism and development methods when making dependency grammars are highly similar to those used at the (lower) levels of morphological disambiguation and shallow syntactic function assignment.

2.1 Possible solutions for Constraint Grammar based treebanking

Given the CG properties described above, in particular the possibility for partial analysis and for linguistically controlled superior precision-recall tradeoff, several strategies for CG-based treebanking are outlined next.

As a common core to them all is the need to specify the necessary minimal recall needed for the application and to create a language model (lexicon and grammars) to meet this required minimal recall (by permitting some level of ambiguity or partial dependency analysis in analyser output). In the content of treebanking, this could mean something like the following:

- morphology: recall of well over 99%.
- syntactic function tagging: recall of 98% or more.
- correctness of syntactic dependency assignment: over 98% of assigned dependency relations should be corrrect.

The amount of unresolved ambiguity or of unattached words resulting from the minimum recall/correctness requirements depends on several factors, e.g. granularity of the grammatical distinctions that the parser operates with; characteristics of the corpora to be analysed; development time available for the grammarian; development/testing methods and resources available; competence/experience of the constraint grammarian. As an educated guess: 20-30% of input sentences might get a complete unambiguous dependency analysis, which means that about three quarters of the sentences retain some ambiguity or receive a partial dependency analysis.

In any case, an important desirable property of this initial effort is that there is no need to revisit the analytic decisions made by the resulting partial CG parser. The main challenge is what to do with the remaining (morphological and functional) ambiguity and words not attached in the dependency structure. Three solutions are next outlined.

2.1.1 Extraction from a partially parsed treebank

To support search of lexico-syntactic structures from text, the simplest solution is to apply the search key only to dependency trees (representing full sentences or sentence parts). As the analyses provided by the parser are as reliable as specified, the extracted patterns will be of sufficient quality for (minor) postprocessing and quantitative analysis. It is also likely that many search patterns will apply to subsentential constructions (that do not need a complete sentence analysis); this means that a much larger part than the above-estimated 20-30% of sentences will be useful for corpus linguistic searches.

A limitation of this approach is that the corpus accessible for linguistic searches will be skewed, as sentence parts outside the coverage of the parser's language model will not be used.

2.1.2 Resolving remaining ambiguity with a hybrid parser

Data-driven statistical parsers are usually trained on hand-annotated treebanks of limited size (thousands or tens of thousands of sentences), and their accuracies (e.g. Labelled Attachment Scores,

LAS), probably fall below the minimum accuracy requirements needed to support linguistic corpus searches (as argued above).

The availability of very large volumes of training data with partial but very dependable morphological and dependency syntactic analyses makes it possible to experiment with training statistical parsing capability to complement (or possibly even replace) partial CG-based parsing, in order to provide a more complete (but still sufficiently accurate) syntactic analysis for text corpora. For instance, it may be the case that lexical information can be used to better advantage in statistical modelling of syntax if the amount of learning data is large (e.g. tens of millions of sentences).

2.1.3 Interactive rule-based dependency parsing

Fully manual syntactic analysis is highly work-intensive. For instance, to provide a dependency analysis and a dependency function to each word in a 20-word sentence, 40 decisions need to be made. This kind of syntactic analysis can easily take several minutes per sentence from a human annotator.

With a high-recall partial dependency parser, probably well over 90% of the analysis decisions are made before there is a need for additional information to support parsing. Given a suitable interface for a human to provide e.g. a part-of-speech disambiguation decision or a dependency analysis to an unattached word in the case of a partially parsed sentence, the language model of the CG parser is usually able to carry on the high-quality syntactic analysis of the sentence, possibly to completion, without further input from the linguist. The reason for this is that the additional analysis provided by the linguist makes the sentence (context) less ambiguous, as a result of which a contextual constraint rule (or a sequence of them) is able to apply, by discarding illegitimate alternative analyses or by adding new dependency relations to the sentence.

The speedup to manual treebanking might be 10-50 fold, which enables cost-effective annotation of much larger treebanks than those available today, but treebanking tens or hundreds of millions of sentences is probably not a practical option even with this semiautomatic method.

3 Ongoing work in FIN-CLARIN

Next we present ongoing work as part of the FIN-CLARIN project (2010-2012) on the creation of a large-scale resource and service for researchers into the Finnish language, focusing on one of its five subprojects, FinnTreeBank. We outline a dependency-syntactic representation for Finnish, and present the first version of the dependency syntactic FinnTreeBank and its use as a "grammar definition corpus" to guide development, testing and evaluation of Constraint Grammar based language models for high-accuracy annotation of large publicly-available Finnish-language corpora, which will be used as empirical data to support linguistic research on Finnish at a large scale.

3.1 Project environment

Our work is done with support from the European CLARIN and METANET consortia, with the following overall aims:

- help researchers discover relevant empirical data and resources more easily with a web service where search is supported e.g. with metadata and persistent identity markers.
- help researchers license and use found resources more easily e.g. with transparent and easy-to-use licensing/access terms and policies.
- help researchers share their own data to support other researchers and to support validation of reported empirical experiments e.g. by means of easy-to-use procedures for data licensing and persistent storage service.
- help researchers use and share existing work by promoting open source.
- help researchers use different resources e.g. by promoting common standards and user-friendly interfaces to data.

At our department, there are several subprojects in the larger META-CLARIN project on different language resources and finite state methods and libraries: Helsinki Finite State Transducer HFST; OMorfi Finnish Open Source Morphology; Finnish WordNet; Finnish Wordbank; FinnTreeBank.

3.2 FinnTreeBank goals and milestones

In addition to th eordinaty academic goal of producing published research with research collaborators, FinnTreeBank has two main goals as a 'producer': (i) to provide large high-quality treebanks of Finnish to the research community; (ii) to provide language models of Finnish as open source for use with open-source software, to help researchers analyse additional texts and to help them modify the language models and/or software for an analysis more suitable for their research question.

Recent and near-term FinnTreeBank milestones include the following:

- Evaluation and selection of language resources, technologies and tools for use in FinnTreeBank developments.
- Initial specification of linguistic representation for initial use in treebanking Finnish, with focus on dependency syntax.
- Manual application of dependency syntactic representation on an initial corpus of 19,000 example utterances from a large descriptive grammar of Finnish (including further specification and documentation of the linguistic representation).
- Subcontracting a 3rd party provider to provide a parsing engine (black box) and automatically parsed treebank (EuroParl, JRC-Aquis) for the web service.
- Development of open-source lexicons, parsing grammars and other resources to support high-quality dependency parsing of Finnish by the research community.
- Delivery of new versions of FinnTreeBank with new corpora and higher quality of linguistic analysis.

3.3 Specifying a grammatical representation with a grammar definition corpus

In order to create a high-quality parser and treebank, we need documentation and examples on the linguistic representation and its use in text analysis. In order to approximate also less frequent structures used in a large corpus of text in a comprehensive and systematic way, we need a maximally exhaustive and systematic set of sentences to be analysed and documented e.g. as a guideline for creating a Parsebank. We propose to use a comprehensive descriptive grammar (typically more than a thousand closely-printed book pages) as a source of example sentences to reach a high and systematic coverage of the syntactic structures in the language. A hand-annotated, cross-checked and documented collection of such a systematic set of sentences – in short, a Grammar Definition Corpus – is a workable initial approximation and guideline for annotating or parsing natural language on a large scale. The initial definitional sentence corpus can be extended with new data when 'leaks' in the grammar/corpus coverage become evident e.g. on the basis of double-blind annotations (Voutilainen and Purtonen 2011).

A result of this effort is a Grammar definition corpus of Finnish, consisting of about 19,000 example utterances extracted from a comprehensive Finnish grammar (Hakulinen at al, 2004), and manually annotated according to a linguistic representation consisting of a morphological description and a dependency grammar with a basic dependency function palette.

We expect use of the Grammar Definition Corpus to have the following benefits:

- A well-documented Grammar Definition Corpus is useful as a guideline for human annotators, to support consistent and linguistically motivated analysis.
- A Grammar Definition Corpus also is useful for one who writes and tests parsing grammars (e.g. in the CG framework): it helps systematic modelling of target constructions, and it also helps document the scope of the language model (what constructs are covered, and what constructions are left outside the scope of the language model).
- Evaluation and testing of language models, corporan and analysers can be done more objectively if the linguistic representation has been specified in a comprehensive and systematic way.
- When annotating new texts e.g. manually, there is a lower chance to come across unexpected linguistic constructions (given the high coverage of the Grammar Definition Corpus), hence less need to redesign or compromise.

- Encountering constructions not covered by the Grammar Definition Corpus is useful data also for writing a more comprehensive descriptive grammar (compared with the original descriptive grammar from which the example utterances were extracted).

To our knowledge, this effort if the first one based on a comprehensive, well-documented set of sentences. The closest earlier approximation to a Grammar definition corpus we know of is an English corpus, tagged and documented in the early 1990's according to a dependency-oriented representation, and consisting of about 2,000 sentences taken from a comprehensive grammar of English (Quirk et al, 1985). However, the Quirk et al grammar contains much more than the 2,000 sentences (i.e. partial coverage in the corpus), and the annotated corpus itself has not been published, though this early effort is briefly described in (Voutilainen, 1997).

3.4 Dependency representation

Our dependency syntactic representation follows common practice in many ways. For instance, the head of the sentence is the main predicate verb of the main clause, and the main predicate has a number of dependents (clauses or more basic elements such as noun phrases) with a nominal or an adverbial function. More simple elements, such as nominal or adverbial phrases, have their internal dependency structure, where a (usually semantic) head has a number of attributes or other modifiers.

The dependency function palette is fairly ascetic at this stage. The dependency functions for nominals include Subject, Object, Predicative and Vocative; adverbials get the Adverbial function; modifiers get one of two functions, depending on their position relative to the head: premodifying constructions are given an Attributive function tag; postmodifying constructions are given a Modifier function tag. In addition, the function palette includes Auxiliary for auxiliary verbs, Phrasal to cover phrasal verbs, Conjunct for coordination analysis, and Idiom for multiword idioms.

The present surface-syntactic function palette can be extended into a more fine-grained description at a later stage; for instance, the Adverbial function can be divided into functions such as Location, Time, Manner, Recipient and Cause. Such a semantic classification is best done in tandem with a more fine-grained lexical description (entity classification, etc).

Sometimes, the question arises whether to relate elements to each other on syntactic or on semantic criteria. As an example from English, consider the sentence "I bought three litres of milk". On syntactic criteria, the head of the object for the verb "bought" is "litres", but semantically one would prefer "milk". Our dependency representation relates elements to each other based on semantic rather than inflectional criteria. Hence our analysis (much as with Prague Tectogrammar and Tree-Bank) gives a dependent role to categories such as conjunctions, prepositions, postpositions, auxiliaries, determiners, attributes and formal elements (formal subject, formal object, etc.). Sometimes this practice creates a conflict with the accustomed notion that there is a certain correspondence between Finnish cases and syntactic functions (e.g. the genitive or partitive case for the object function): for instance a premodifying quantifier may have the genitive case (for objects), while the semantic object's case may follow from the valency structure of the quantifier. – This feature, like many others, needs to be taken into account in the design of a corpus linguist's search/extraction interface.

3.5 Sample analyses

In this section, some example sentences from the grammar definition corpus are shown in visual form to illustrate the dependency representation outlined above.

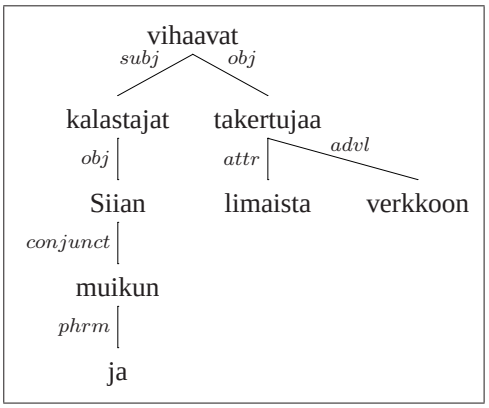
3.5.1 Clausal premodifiers

In Finnish, nominals can have clausal modifiers on both sides (premodifying and postmodifying positions). For instance, premodifying participles can have verbal arguments of their own. For instance, the participle "muistuttavia" acts as a premodifier of the noun "kissannaukujaisia" but has also an object, "glissandoja", as its dependent.

kissannaukujaisia
[cat-meowings.PartitivePlural] muistuttavia

[resembling.Pcp] glissandoja [glissandos.Plural]

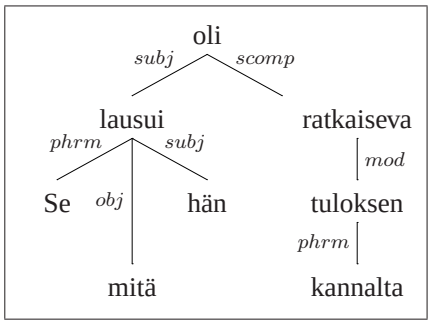
We have also described a restricted class of nouns like this. For instance, agentive nouns like "kalastajat" (fishers) can have objects like "siian" (whitefish) in a premodifying position:



Siian [whitefish.GenSg] ja [and] muikun [vendace.GenSg] kalastajat [fisher.NomPl] vihaavat [hate.VPres] limaista [slimy.PartSg] verkkoon [net.IllatSg] takertujaa [clinger.PartSg].

3.5.2 Phrase markers

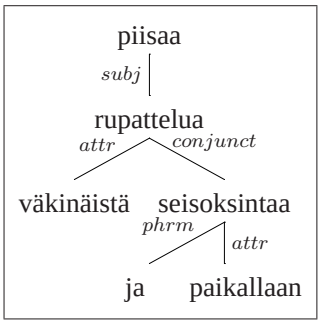
Formal ”se” (’it’) is described as a phrase marker for the subject clause "mitä hän sanoi" (what s/he said); likewise the postposition "kannalta" (regarding) is described as a phrase marker of the noun "tuloksen" (result):



Se [it.NomSg] mitä [what.PartSg] hän (s/he.NomSg) lausui [said] oli [was] tuloksen [result.GenSg] kannalta [regarding.Postposition] ratkaiseva [decisive.NomSg].

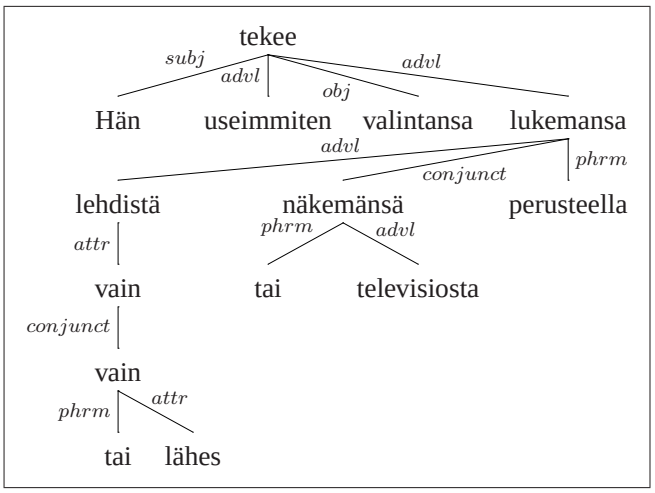
3.5.3 Coordination

The conjunction "ja" (and) is described as a phrase marker of the following conjunct "paikallaan seisoksintaa" (steady standing), which in turn is described as coordinated dependent of the preceding conjunct "väkinäistä rupattelua" (forced chatting):



väkinäistä [forced.PartSg] rupattelua [chatting.PastSg] ja [and] paikallaan [steady.AdessSg] seisoksintaa [standing.PartSg] piisaa [suffices].

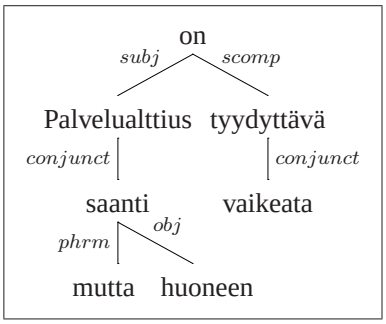
Here is an example with multiple coordinations. The attributes "vain" (only) and "lähes vain" (almost only) are coordinated with "tai" (or); the participles "lukemansa" (read) and "näkemänsä" (seen) are coordinated also with "tai":



Hän [s/he] tekee [makes] useimmiten [usually] valintansa [choice.GenPl] vain [only] tai [or] lähes [almost] vain [only] lehdistä [newspaper.ElatPl] lukemansa [read.PcpPoss] tai [or] televisiosta [television.ElatSg] näkemänsä [see.PcpPoss] perusteella [on-the-basis-of.Postposition].

3.5.4 Ellipsis

Two clauses are coordinated: S-V-C with S-C (verb missing). The subject of the elliptical clause ("huoneen saanti") is described as a conjunct of the subject of the first clause ("palvelualttius"), and the predicative complement ("vaikeata") is described as a conjunct of the predicative complement of the first clause ("tyyydyttävä"):



Palvelualttius [service-readiness.NomSg] on [is] tyydyttävä [satisfactory.NomSg], mutta [but] huoneen [room.GenSg] saanti [getting.NomSg] vaikeata [difficult.NomSg].

4 Ongoing developments

In this final section, we describe some ongoing or near-term developments to meet the objectives of the FinnTreeBank project during the next year and a half.

4.1 Harmonisation of morphology with syntax

The initial dependency syntactic annotation (function and relation assignment by linguists) was mainly done independently of morphological analysis. One motivation for this is savings in labour: a morphological description designed before a syntactic description usually needs to be revised when the detailed decisions on how to model syntax are made (which means that also morphological annotations require substantial revisions). In our solution, the morphological description can be designed "at one go" to agree with the documented syntactic representation. A further advantage of our solution is that resolution of morphological ambiguities can be done with the help of available higher-level (syntactic) analysis.

In practise, the morphological and lexical analysis will be based on the Omorfi open-source lexical and morphological language model (partly derived from publicly available word lists by the Finnish Research Centre of Domestic Languages) and finite-state (HFST) analysis tools. Along with this semiautomatic synchronisation/tagging effort, also consistency checks and corrections to syntactic annotation can be made to improve the quality of the grammar definition corpus treebank.

The morphologically synchronised treebank will be delivered in CONLL-X form with extensive documentation to enable e.g. development of statistical language models for parsing.

4.2 Dependency treebank and parser engine by third-party provider

Another ongoing development is done by a third-party provider (Lingsoft and its collaborators, the Turku BioNLP Group at University of Turku) who is building a statistical language model for dependency parsing on the basis of the initial grammar definition corpus with the dependency syntactic annotation. On the basis of the contract, the provider will deliver automatically parsed language resources (EuroParl corpus and JRC-Aquis, totalling tens of millions of words of Finnish) for distribution via the FIN-CLARIN service.

The provider will also provide a licence to the executable parser engine to enable annotation of additional corpora for FIN-CLARIN users.

4.3 Development of open-source language models for dependency parsing

Alongside the above developments, the FinnTreeBank project develops open-source language models using open-source tools and development environments (e.g. HFST morphology and syntax, VISL cg3) for dependency parsing of Finnish. The FIN-CLARIN users will benefit from the open-source development as it enables them to adapt and apply the language models and resulting parsers to better answer their research questions and to better support development of e.g. Artificial Intellignce solution prototypes. The results of this development can also be used for providing an alternative annotations to existing and new corpora (treebanking).

Also development of commercial or open-sector web services and other solutions should benefit from availability of open-source language technological tools and resources.

4.4 Experiments on treebanking methods

When initial versions of the language models mature, it will be possible to start experimenting with alternative treebanking methods outlined above in section 2.1. This research will likely be carried out in collaboration with other research teams towards (and hopefully after) the end of the ongoing project. The results of the experiments will provide guidance on treebanking efforts in the longer term in Finland, and hopefully in other projects as well.

Acknowledgments

The ongoing project has been funded via CLARIN, FIN-CLARIN, FIN-CLARIN-CONTENT and META-NORD by EU, University of Helsinki and the Academy of Finland. I wish to thank Mikaela Klami, Tanja Purtonen, Satu Leisko-Järvinen, Kristiina Muhonen, Tommi Pirinen and Sam Hardwick, as well as other HFST team members, for their support of this project.

References

Eckhard Bick. 2000. *The parsing system Palavras*. Aarhus: Aarhus University Press.

Christer Samuelsson and Atro Voutilainen. 1997. Comparing a linguistic and a stochastic tagger. *Proc. EACL-ACL'97*.

Pasi Tapanainen and Timo Järvinen. 1997. A non-projective dependency parser. *Proceedings of the 5th Conference on Applied Natural Language Processing*. Washington, D.C.

Auli Hakulinen, Maria Vilkuna, Riitta Korhonen, Vesa Koivisto, Tarja Riitta Heinonen and Irja Alho. 2004. *Iso suomen kielioppi* [Large Finnish Grammar]. Helsinki: Suomalaisen Kirjallisuuden Seura. Online version: http://scripta.kotus.fi/visk URN:ISBN:978-952-5446-35-7.

Katri Haverinen, Filip Ginter, Veronika Laippala, Tapio Viljanen, Tapio Salakoski. 2009. Dependency Annotation of Wikipedia: First Steps towards a Finnish Treebank. *Proceedings of The Eighth International Workshop on Treebanks and Linguistic Theories (TLT8)*.

Matthias Kromann. 2003. The Danish Dependency Treebank and the underlying linguistic theory. *Proc. of the TLT 2003*.

Krister Lindén, Miikka Silfverberg and Tommi Pirinen. 2009. HFST Tools for Morphology – An Efficient Open-Source Package for Construction of Morphological Analyzers. *Proceedings of the Workshop on Systems and Frameworks for Computational Morphology 2009*, Zürich, Switzerland.

Marie Mikulova, Alevtina Bemova, Jan Hajic, Eva Hajicova, Jiri Havelka, Veronika Kolarova, Lucie Kucova, Marketa Lopatkova, Petr Pajas, Jarmila Panevova, Magda Razimova, Petr Sgall, Jan Stepanek, Zdenka Uresova, Katerina Vesela, and Zdenek Zabokrtsky. 2006. Annotation on the Tectogrammatical Level in the Prague Dependency Treebank. Annotation Manual. Technical Report 30, UFAL MFF UK, Prague, Czech Rep.

Joakim Nivre, Jens Nilsson and Johan Hall. 2006. Talbanken05: A Swedish Treebank with Phrase Structure and Dependency Annotation. *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC2006)*.

Ville Oksanen, Krister Lindén and Hanna Westerlund. 2010. Laundry Symbols and License Management: Practical Considerations for the Distribution of LRs based on experiences from CLARIN. *Proceedings of the seventh international conference on Language Resources and Evaluation (LREC2010)*.

Ted Pedersen. 2008. Last Words: Empiricism Is Not a Matter of Faith. *Computational Linguistics, Volume 34, Number 3, September 2008*.

Randolph Quirk, S. Greenbaum, G. Leech, and J. Svartvik. 1995. *A comprehensive grammar of the English language*. London: Longman.

Atro Voutilainen, Krister Lindén and Tanja Purtonen (forthcoming). 2011. Designing a Dependency Representation and Grammar Definition Corpus for Finnish. *Proc. CILC 2011 - III Congreso Internacional de Lingüística de Corpus*.

Atro Voutilainen. 1997. Designing a (Finite State) Parsing Grammar. Roche and Schabes, Eds, *Finite State Language Processing*. The MIT Press.

An Efficient Constraint Grammar Parser based on Inward Deterministic Automata

Anssi Yli-Jyrä
The Department of Modern Languages, PO Box 24, 00014 University of Helsinki, Finland
anssi.yli-jyra@helsinki.fi

Abstract

The paper reconceptualizes Constraint Grammar as a framework where the rules refine the compact representations of local ambiguity while the rule conditions are matched against a string of feature vectors that summarize the compact representations. Both views to the ambiguity are processed with pure finite-state operations. The compact representations are mapped to feature vectors with the aid of a rational power series. This magical interconnection is not less pure than a prevalent interpretation that requires that the reading set provided by a lexical transducer is magically linearized to a marked concatenation of readings given to pure transducers. The current approach has several practical benefits, including the inward deterministic way to compute, represent and maintain all the applications of the rules in the sentence.

1 Introduction

Constraint Grammar (CG) (Karlsson et al., 1995) is a text parsing method with benefits over statistical methods: a low memory footprint, run-time speed, linguistic detail, data bootstrapping, incremental development, and applicability to linguist's needs. Despite its wide use, the common understanding about its algorithms is still shallow. The current work attempts to reduce this gap.

1.1 The Background

The dawn of CG was marked by a number of related developments. Some resource sensitive parsers (Marcus, 1980; Krauwer and des Tombe, 1981; Church, 1988; Blank, 1989) had started to simplify over the parsers based on augmented transition networks (Woods, 1970). The Taggit program (Greene and Rubin, 1971; according to Tapanainen 1999) was an early context-dependent tagger. Some generative grammars were related to automata and local constraints both in syntax (Peters and Ritchie, 1969; Joshi and Levy, 1982), and in phonology (Johnson, 1972; Koskenniemi, 1983). Their constraints were similar to local grammars that describe word chain characteristics (Gross, 1968; Maurel, 1989; Mohri, 1994). The systems of hard constraints gave rise to consistency enforcing methods (Huffman, 1971; Barton, Jr., 1986; Maruyama, 1990).

The CG framework (Karlsson et al., 1995) applies disjunctively ordered rules iteratively to implement a system of soft constraints. Some CG parsers have been described (Karlsson, 1990; Tapanainen, 1996; Graña et al., 2003; Didriksen, 2010; Peltonen, 2011; Hulden, 2011).

The several later parsing methods bear similarities to CG. These include Finite-State Intersection Grammar (FSIG) (Koskenniemi et al., 1992), cascade parsing and chunking (Joshi and Hopely, 1996; Abney, 1991; Grefenstette, 1999), replace rule sequences (Karttunen, 1997; Aït-Mokhtar and Chanod, 1997), lenient composition (Karttunen, 1998), voting constraints (Oflazer and Tür, 1997), error-driven parsing (Brill, 1992; Lager, 2001), bimachines (Roche, 1994; Skut et al., 2004; Peikov, 2006), iterated finite transducers (Roche, 1997b; Bordihn et al., 2006), restarting automata and contextual grammars (Plátek et al., 2003; Jurdziński et al., 2005), and logic programming (Lindberg and Eineborg, 1998; Lager and Nivre, 2001).

Throughout the paper, the discussion is made more concrete by experiments on a version of the Finnish Constraint Grammar (FINCG), a freely available rule set developed originally for Finnish by Fred Karlsson in Helsinki. The preliminary experiments merely suggest the rough degrees of cardinality in various aspects of processing complexity.

1.2 The Contributions

The current independent work of the author (Yli-Jyrä, 2010) aims at reconceptualizing CG parsing in the framework of finite automata. It describes an efficient parsing procedure where the local ambiguity is summarized with feature vectors. Moreover, a nearly complete CG rule compiler and a partially implemented parser are briefly reported.

The presented approach is in strict contrast to some prior parsers where the local ambiguity domains are represented, tested and reduced by manipulating a linear representation of the set of readings. The linear representation gives rise to CG parsing as transducer sequences (proposed by Lauri Karttunen; see Voutilainen 1994:39, Koskenniemi 1997, Peltonen 2011, Hulden 2011), but is likely to become a bottleneck if syntactic functions and argument structures are provided in the input. In contrast, the current proposal eliminates the distinct syntactic disambiguation rules and compacts the representation of local ambiguity domains.

The current work uses pure finite-state automata that are described, at a high level, using rational sets and series. On the other hand, the paper involves schematic string matching and bidirectional memoization when intersecting automata. These low-level techniques are efficient but differ from standard sequential processing models.

The paper transfers some techniques from the author's prior research on FSIG parsing to the CG framework: (1) *Indexing the transition labels of a template automaton* will compress the implementation of reading subsets (Yli-Jyrä, 1995). (2) The *split languages* of the form $L \subseteq \Sigma^*\Delta\Sigma^*$ (Σ, Δ disjoint alphabets) will represent context conditions (Yli-Jyrä, 2011a). (3) The *position-wise flag diacritics* (Yli-Jyrä, 2011b) will postpone the computation of negation in negative contexts. (4) On-the-fly *inward determinization* (Yli-Jyrä, 2010; Yli-Jyrä, 2011a) will facilitate the iterated computation of product automata. (5) *Preference relations* (Yli-Jyrä, 2007) will enable the ordering of rules and their potential applications. (6) The *infiltration* operation (Sakarovitch, 2009) — a natural implementation of *simple multitape automata* (Yli-Jyrä, 2005) — will elegantly compile the rule conditions. (7) The *string schemas* (Yli-Jyrä, 1995; Yli-Jyrä, 2005; Yli-Jyrä, 2011b) will reduce the length of paths in automata. It is expected that the started CG implementation effort will produce ideas that will reciprocally enrich the FSIG framework.

2 The Primary Representation

The Tokens The natural language sentence – the input of the parser – is preprocessed for parsing by segmenting it into *orthographic words* aka *tokens* $t_1, ..., t_n$. Thus, we obtain e.g. the orthographic words `"<It>"`, `"<rains>"`, `"<.>"` from the sentence "*It rains.*" for the lexical look-up. The resulting segmentation must be unique.

The lexicon is a regular relation that relates the orthographic words with strings consisting of lexemes, morphological labels, syntactic labels and semantic labels – we will call all these symbols naively just *tags* and their strings *readings*.

For every token t_i ($1 \leq i \leq n$), the lexicon provides a set of token-analysis pairs $(t_i, a_{i,j})$ that are linearized to a set of strings $t_i a_{i,j}$ as in (1) — in general, such linearization is not a regular relation, and it would be purer to drop the orthographical word in the analysis. Nevertheless, the set is not converted to a linear string in the current work.

$$\left\{\begin{array}{l}\texttt{"<muuta>" "muu" Q PRON PTV SG}\\ \texttt{"<muuta>" "muuttaa" V IMPV ACT SG2}\\ \texttt{"<muuta>" "muuttaa" V PRES/IMPV ACT NEG}\end{array}\right\} \quad (1)$$

The Cohort Automata The purpose of the CG parser is to reduce excessive readings through removals and selections. By a metaphor, the group of readings for each token is called a *cohort* and the readings manipulated by the operations are called *targets*.

The current parser contains a custom input function for cohorts. This produces deterministic acyclic finite automata $c_1, ..., c_n$, called *cohort automata* (Figure 1), and minimizes them. The cohort automata constitute the *primary representation* of the cohorts.

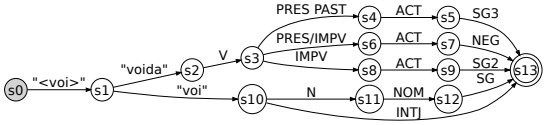


Figure 1: A cohort automaton.

The operators of the CG rules (Tapanainen, 1996) define how cohort automata change if the rules are applied to them. Each operator is a regular relation. To avoid non-termination (Didriksen, 2010), we exclude all rule operators that replace readings, shorten the readings (e.g. by removing `@`-tags), and introduce readings. This leaves such

operators that refine the cohort strictly monotonically: readings are removed with the REMOVE and SELECT operators and new (irreversible) distinctions are introduced with the ADD operator.

The Template Automaton A common experience is that the interface between the grammar and the morphological component gets easily broken if one component is changed. It is thus desirable that CG grammars are augmented with a specification that describes the possible readings by a *template automaton*, a simple positional model (a linear finite automaton) for the tags.

The grammar distinguishes only a finite number of different kinds of tags. In this sense, the tags (the tag types) form a finite set, T. The interface of FINCG needs some 1230 tags, slightly more than are actually used by the rules.

A template automaton for the FINCG is given by a regular expression (2). The special symbol 0 is interpreted as the empty string. This gives an automaton with 1272 transitions (removing the symbol 0 results in 3853 transitions).

$$
\begin{aligned}
&(\texttt{"<>"}|\texttt{"<ajan>"}|\ldots)\ \texttt{0}\ (\texttt{"<(.*)ja>"r}|\ldots)^*\ \texttt{0}\\
&(\texttt{""}|\texttt{"aamu"}|\ldots)\ \texttt{0}\ (\texttt{DV-JA}|\ldots)^*\ \texttt{0}\ (\texttt{0}|\texttt{DEM}|\ldots)\\
&(\texttt{A}|\texttt{V}|\ldots)\ (\texttt{0}|\texttt{PRES}|\ldots)\ (\texttt{0}|\texttt{ACT}|\ldots)\ (\texttt{0}|\texttt{SG1}|\ldots)\\
&(\texttt{0}|\texttt{CMP}|\ldots)\ (\texttt{0}|\texttt{ALL}|\ldots)\ (\texttt{0}|\texttt{SG}|\ldots)\ (\texttt{0}|\texttt{P-3}|\ldots)\\
&\quad(\texttt{ko}|\texttt{han}|\ldots)^*\texttt{0}\ \ (\texttt{0}|\texttt{@ADVL}|\texttt{@SUBJ}|\ldots). \qquad (2)
\end{aligned}
$$

The Dynamic Aspects The expression (2) uses two meta-symbols: "<>" matches any orthographical strings such as "<foo>", and "" matches any lexemes. The tags such as "<(.*)ja>"r are inserted by the input function to the reading strings when the regular expression in it matches the orthographical string such as "<opettaja>"r.

When the parser is used, the conformance of the readings against the model (2) is checked on the fly. The reported anomalies in the lexicon can then be fixed in order to optimize the interoperability between the lexicon and the grammar. In addition, the tags in all readings are indexed with the corresponding source states in the template automaton. The tag DEM, for example, thus becomes $\texttt{DEM}_7$. There are tags that can correspond to several states — especially if the template automaton is without the symbol 0.

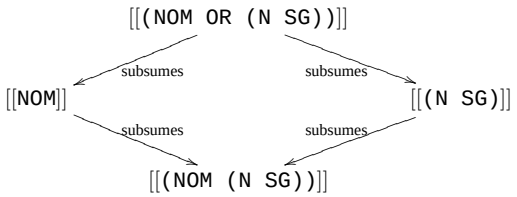


Figure 2: An excerpt of the Boolean lattice.

3 Testing the Cohorts

3.1 The Features of Readings

The CG-2 rules refer to the abstract *features* of complete readings through tags, combined tags, lists, and sets (Tapanainen, 1996) — let us call them *set definitions*. Each set definition α has the denotation $[[\alpha]]$ that is a subset of the readings recognized by the template automaton. Two definitions are *equivalent* if their denotations coincide. The special expression $(\texttt{*})$ subsumes the whole universe while expression $(\texttt{N SG})$ subsumes all Singular Noun readings.

In the VISL CG-3 (Didriksen, 2010) system and the current system, the order of tags does not matter i.e. $[[(\texttt{N SG})]] = [[(\texttt{SG N})]]$. The denotationally equivalent set definitions form, thus, a Boolean lattice (Figure 2) with denotational union (OR), denotational intersection ( ), and denotational complement (\) operations. The implementation of these operations benefits from the restrictions imposed by the template automaton.

The denotation $S \subseteq T^*$ of every set definition, i.e. a feature f, gives rise to a weighted finite automaton (Sakarovitch, 2009) that recognizes its characteristic series χ_f: $T^* \to \mathbb{N}$ defined by $\chi_f(x)$=1 for all $x \in S$ and $\chi_f(x)$=0 otherwise. We will call this automaton a *feature automaton*. A simple expression, (V), compiles into an automaton with more than 1100 (logical) transitions. To test a feature f against a reading $r \in T^*$, we simply compute $\chi_f(r)$ with the automaton.

When extended to the set of all features F in the grammar, the feature automaton recognizes a series $T^* \to \mathbb{N}^F$ whose coefficients are integer vectors. The easiest implementation of this would be the union of $|F|$ feature automata. The size of this automaton is a severe problem. In the FINCG rule set, the set of tags (T) and the set of features (F) have roughly the same cardinality (respectively: 1133, 1216). Thus, more than a million (logical) transitions are traversed to check all the readings and features in the worst case (consider a cohort

automaton that equals the template automaton). A minimal deterministic automaton is not likely to be any better solution as the deterministic union of all feature automata requires, in the worst case, $O(2^{|F|})$ different final states.

To reduce the complexity of the deterministic union of feature automata, the parser uses a contraction technique (Yli-Jyrä, 1995; Yli-Jyrä, 1997; Roche, 1997a) that hides transitions that are irrelevant to feature recognition. The resulting deterministic automaton will match indexed subsequences in the readings. Given that N and GEN are labels that occur respectively in states 8 and 13 of the template automaton, the set definition (N \GEN) corresponds to a 3-state automaton that recognizes the subsequences

$$\begin{aligned}\mathtt{N}_8(&\mathtt{0}_{13}|\mathtt{ABE}_{13}|\mathtt{ABL}_{13}|\mathtt{ACC}_{13}|\mathtt{ADE}_{13}|\mathtt{ALL}_{13}|\mathtt{CMT}_{13}|\\ &\mathtt{ELA}_{13}|\mathtt{ESS}_{13}|\mathtt{ILL}_{13}|\mathtt{INE}_{13}|\mathtt{INS}_{13}|\mathtt{LAT}_{13}|\mathtt{LOC}_{13}|\\ &\mathtt{MAN}_{13}|\mathtt{NOM}_{13}|\mathtt{PTV}_{13}|\mathtt{TRA}_{13}).\end{aligned} \quad (3)$$

3.2 The Features of Cohorts

In general, the cohorts may mix both readings that satisfy a given feature and readings that do not satisfy it. Therefore, the features become 3-valued at the cohort level: they can be *positive* (+), *negative* (-) or *ambivalent* (?). Testing the cohorts corresponds to computing the function: $2^{T^*} \rightarrow \{+,-,?\}^F$, where the domain 2^{T^*} contains all the possible cohorts and the range $\{+,-,?\}^F$ contains all combinations of the features.

For each cohort, the truth-value of a particular feature f is determined by counting the number of true readings in the domain of χ when it is restricted to the set of readings S in the cohort automaton: the feature automaton is first restricted with the cohort and then the sum of the successful paths is computed e.g. by replacing all the inputs with the empty string. Thus, we compute $\sum_{x \in S} \chi_f(x)$.

In order to get the correct counts, the feature automata must be *path unambiguous* i.e. they assign, at the most, one successful path to each tag sequence. Ensuring an unambiguous automaton for set definitions like (C OR "että") requires special attention, because the features C and "että" are non-exclusive and are separately true in the reading ("<että>" "että" SUB C).

Finally, the integer vectors $\mathbb{N}^F$ are mapped to the vectors of (3-valued) truth values, $\{+,-,?\}^F$. If the count of a feature equals the cardinality of the cohort, the feature is positive +. Otherwise, it is negative - if the cardinality is zero or ambivalent ? in other cases.

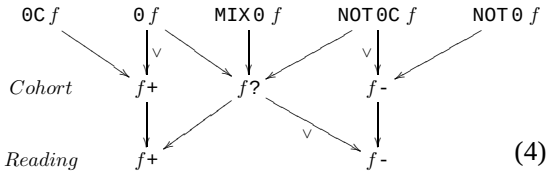
Simple Conditions A typical CG rule specifies three things: an *operation* that tells how a cohort will change if the rule is applied to it, the *context specification* that tells where the rule can be applied, and a *target* that complements the operation and the context specification. The context specification is a conjunction of simple, possibly linked, context conditions (Tapanainen, 1996).

A simple context condition in CG rules is, formally, a 6-tuplet ⟨*origin*, *polarity*, *position*, *mode*, *set definition*, *barrier condition*⟩ where

- the *origin* refers to the target cohort (nothing) or to the hit of the previous positive condition (LINK)
- the *polarity* is either *negative* (NOT), *positive* (nothing), or *ambivalent* (MIX)
- the *position* is either an *absolute position* (@1, @2, ...), a *relative position* (..., -2, -1, 0, 1, 2, ...) with 0 as the origin, or an *unbounded set of relative positions* outwards from the origin (..., *-2, *-1, *0, *1, *2, ...)
- the *mode* is *careful* (C) or *normal* (nothing)
- the *barrier condition* is either none (nothing) or a set α (BARRIER α).

A condition such as ⟨LINK, not, 0, C, N, BARRIER CLB⟩ is written simply as (LINK NOT 0C N BARRIER CLB). The reader is referred to CG manuals for a complete description of context conditions (Karlsson et al., 1995; Tapanainen, 1996).

The combination of polarity and mode tells how the cohort's feature-values are converted back to Boolean truth-values needed in the grammars. This is illustrated in (4).



(4)

The core CG grammar rules — those whose operator is SELECT and REMOVE — specify a set definition called a *target*. A target (NOM) is, in fact, shorthand for condition (MIX 0 NOM). The condition matches cohorts where some readings contain the NOM tag and some do not.

4 The Secondary Representation

We will now see how the conjunction of the contextual conditions is represented in the parser.

4.1 Marking the Potential Applications

The Feature-Value Alphabet Internally, the set definitions are mapped to feature numbers that form the feature alphabet F. The numbers of equivalent set definitions coincide. For example, the current parser assigns the feature numbers 31 and 39 to set definitions $(\texttt{PTV})_{31}$ and $(\texttt{"yhtään" ADV})_{39}$, respectively. We write `31.PTV` when we emphasize the feature number (the key) rather than the set definition (the legend).

The core FINCG rules mention 1216 distinct features (F) and 1813 feature-value pairs such as (`5.NOM`,`+`) and (`11.SG`,`?`).

The *secondary representation* for the cohorts consists of the feature-value pairs ($P{=}F{\times}\{$`+`, `-`, `?`$\}$) and cohort boundaries ($\bullet$). The pairs in P are written simply as `5.NOM+` and `11.SG?`. The whole sentence is represented by the $\bullet$-marked concatenation of the cohort-wise lists $c'_1, ..., c'_n$ of $|F|$ numerically ordered feature-value pairs.

$$w = c'_1 \bullet c'_2 \bullet ... \bullet c'_n. \tag{5}$$

The Rule Marker Alphabet In order to talk about rules and where they apply, we define a symbol alphabet for the rules. First, the rules are numbered. The rule compiler assigns numbers 343 and 865 to the rules

$$\texttt{SELECT}(\texttt{"yhtään" ADV})_{39} \quad \texttt{IF}(\texttt{NOT}^{*}1\,(\texttt{PTV})_{31}); \tag{343}$$

$$\texttt{SELECT}(\texttt{N SG})_{821} \quad \texttt{IF}(\texttt{1*}\,(\texttt{V SG3})_{820}\ \texttt{BARRIER}(\texttt{CLB})_{3}) \quad (\texttt{NEGATE*-1}(\texttt{N NOM})_{715}\texttt{BARRIER}(\texttt{CLB})_{3}). \tag{865}$$

For every rule, there are two *rule markers* that are used to indicate the satisfied context conditions of the rules. The markers form the set $R = \{$`@1.`r`@`, `@2.`r`@` $\mid r$ is a rule in the grammar $\}$. As to FINCG, there are about 1380 `SELECT`/`REMOVE` rules and 22 (1,6%) of them use both kinds of markers.

The marker `@1.865@` will be used to indicate the cohorts whose contexts satisfy the licensing context conditions of the rule 865. The marker `@2.865@` will be used to indicate the cohorts that satisfy a prohibiting context condition. The prohibiting context conditions are stronger and will prevent the application of the rule even if the licensing context condition is true (Yli-Jyrä, 2011b).

The CG-3 syntax for rule conditions separates the cohort-internal negation (`NOT`) from the global negation (`NEGATE`) (Didriksen, 2010). The first refers to negative features and the second starts a prohibiting condition according to the CG rule syntax. For example, the rule 343 selects the Adverb (`39.ADV`) reading if the next word has no Partitive (`31.PTV`) readings. In contrast, the rule 865 selects the Nominative Singular (`821.(NOM SG)`) reading if the cohort is followed by potential (i.e. positive or ambivalent) 3rd Person Singular Verb (`820.(V SG3)`) within the same clause (`BARRIER (3.CLB)`) unless (`NEGATE`) the cohort is followed by potential Nominative Noun (`715.(N NOM)`) cohort within the same clause.

Split Languages The string w provides a matrix structure against which the licensed and prohibited applications of CG rules are indicated: for each cohort where the rule r is licensed, we produce a copy of w and insert the marker `@1.`r`@` into the end of the cohort's feature-value list in this copy. The marker `@2.`r`@` is inserted in a similar way to other copies that indicate prohibited contexts.

Let $\Sigma = P \cup \{\bullet\}$. A language of the shape $L \subseteq \Sigma^* R \Sigma^*$ is called a *split language*[1]. To facilitate the formal account of the marking, define a mapping $h : \Sigma^* R \Sigma^* \to \Sigma^*$ by $h = \{(vxy, vy) \mid v, y \in \Sigma^*, x \in R\}$. The inverse of the image of w is $h^{-1}(w)$, the language of the possible ways to insert a rule marker into the string w. This language is recognized by an automaton that is very similar to the linear automaton recognizing the string w. The automaton has $O(n|F|)$ states and $O(n|F||R|)$ transitions.

The union of the licensing and prohibiting contexts of each rule r forms a split regular language $C_r \subseteq \Sigma^* R_r \Sigma^*$ where $R_r = \{$`@1.`r`@`, `@2.`r`@`$\}$. For the rule, the marked copies of the string w are obtained as the intersection $W_r = C_r \cap h^{-1}(w)$. The potential applications of all rules are obtained as the union $W = \bigcup_{r \in R} C_r \cap h^{-1}(w)$.

4.2 Controlling the Application Order

The prohibited applications are subtracted from the licensed ones by computing $W' = \{v$`@1.r@`$y \mid v$`@1.r@`$y \in W, v$`@2.r@`$y \notin W\}$. Such preference restrictions can be implemented with a matching

[1] The term was proposed to me by J. Sakarovitch.

method due to Dale Gerdemann and Gertjan van Noord (see Yli-Jyrä, 2007, 2011b). The same method implements application order modes.

The current system is flexible enough to support any standard application mode. Normally, the application of an earlier rule in the grammar is preferred over the later rules. On the other hand, it is psycholinguistically motivated to process the sentence from left to right. By emphasizing the leftmost position, for example, we get the following restriction of W':

$$\begin{aligned} W'' = \{ & v\texttt{@1.r@}y \in W' \mid \\ & \neg\exists u\texttt{@1.r@}z \in W' \text{ s.t. } vy = uz, |u| < |v| \\ & \neg\exists v\texttt{@1.q@}y \in W' \text{ s.t. } q < r\}. \end{aligned} \quad (6)$$

The obtained set W'' contains (at the most) one marked copy w' of the sentence w. The marker in this copy indicates which rule is applicable to which cohort.

When given an m-state deterministic automaton representing the set W, we can compute W' and W'' in $O(m)$ time. It should be noted that the subsets of $\{vxy \mid vy = w, x \in R\}$, where w is the sentence and R is the rule marker alphabet, are always regular languages and their recognizers are minimizable in linear time.

5 The Context Automata

For each rule $r \in R$, the intersection $C_r \cap h^{-1}(w)$ is computed through the well-known product construction (see Sakarovitch, 2009).

The constructed product may have a large number of states: Let the context language C_r be given by an $O(m)$-state deterministic finite automaton. The minimal automaton recognizing the language $h^{-1}(w)$ has $O(n|F|)$ states. The product of these automata has, thus, $O(nm|F|)$ states. When this is repeated for all $r \in |R|$ rules, the total state complexity is $O(nm|R||F|)$.

The product is potentially constructed several times. A pathologically ambiguous cohort can be refined separately by $|R|$ rules if the targets of the rules have disjoint denotations. Therefore, the computation of the intersection is iterated $O(n|R|)$ times in the worst case. This means that the total time complexity of the context testing is $O(n^2m|R|^2|F|)$ in the worst case.

We need optimizations that reduce the effects of (1) the context states, (2) the iterations, (3) the features, and (4) the parallel rules. The following will sketch some important optimization strategies.

5.1 Reducing the Effect of Context States

The Baseline The deterministic automaton recognizing the context language C_r is deterministic up to point where an R-transition is followed. After this point, the product automaton contains, in the worst case, $O(m)$ parallel paths (Figure 3(i)).

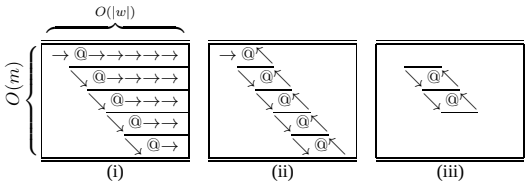


Figure 3: The benefits of inward processing.

Inward Processing It is desirable to find a method that computes the product or it restriction W'' in time that does not depend on the state complexity of the context languages. Three solutions that fulfill this condition are available:

1. The function returning W'' for each w is a has a deterministic realization using *deterministic bimachines* (Skut et al., 2004; Roche, 1994; Peikov, 2006; Hulden, 2011).

2. The context language C_r is recognized by an *inward deterministic automaton (IDA)* (Yli-Jyrä, 2011a). The approach allows for greater flexibility in the application ordering.

3. An IDA can be factorized to possibly smaller left- and right-sequential transducers (Roche, 1997a). Nondeterministic bimachines could be used too (Santean and Yu, 2006).

An *IDA* is a nondeterministic automaton that recognizes a split language $L \subseteq \Sigma^* R \Sigma^*$ and is deterministic for all prefixes in Σ^* and codeterministic for all suffixes in Σ^*. When intersected with the linear automaton representing the language $h^{-1}(w)$, the restricted product approaches the R-transitions deterministically from both sides (Figure 3(ii)). As a result, each of $O(n)$ positions in w corresponds, at the most, to two states in the restricted product (Figure 4). With one IDA per rule, the total time complexity is bounded by $O(n^2|R|^2|F|)$.

State Explosion It would be nice to construct a combined IDA that recognizes the split language $\cup_{r \in R} C_r$. This would reduce the total computation time to $O(n^2|R||F|)$ if we assumed that parallel

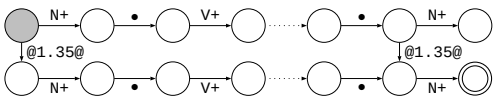


Figure 4: An inward deterministic product.

marker transitions are inserted into the result in $O(1)$ time. However, the whole grammar cannot be combined to an IDA in general. Firstly, there are simple context conditions that require a very large number of IDA states. Secondly, the size of a combined IDA would be $O(m^{|R|})$ if each rule-specific IDA had m states at the most.

5.2 Reducing the Effect of Iterations

Dynamic Programming The product automaton is not constant, but reflects the changes in the sentence w when rules are applied. With inward deterministic contexts, the product is easier to keep up to date. Namely, the previously computed product automaton can be refreshed locally around the changed cohort (Figure 3(iii)).[2] If the distance between the successive refinements is $O(1)$ according to the amortized analysis over the total of $O(n|R|)$ refinements, the local refreshing of the product improves the time complexity to $(n|R|^2|F|)$.

On-the-fly Inward Determinization When computing the intersection $C_r \cap h^{-1}(w)$ with a nondeterministic recognizer for C_r, the result can be determinized easily on-the-fly due to the special structure of the language $h^{-1}(w)$. The result is still virtually deterministic from both sides, allowing for the efficient refreshing of the product after a cohort is refined and local changes in w occur. Under the assumption of $O(1)$ distance between the refinements, the worst case time complexity is now $O(nm|R|^2|F|)$.

5.3 Reducing the Effect of Features

The Path Length Problem A problem with all intersection methods is that they process full-length paths. The feature values in each cohort are read one-by-one, resulting in many states and transitions. Because the length of the string w is $O(n|F|)$, the path length is a practically significant problem.

[2]The dynamics of the optimal refreshing of the product has been studied by the author in a PSC submission (2010).

Contracted Contexts To address the path length problem, the rule compiler of the parser contracts the strings in a context language C into the patterns in a contracted context language C' that retains just the necessary details. In the patterns, the cohort boundaries are retained, while the feature-value symbols in F are retained only where the rule condition refers to them. For example, the condition (`-5 C`) corresponds to the regular language $\bullet^*$`C`$\bullet\bullet\bullet\bullet\bullet$`@1.`$r$`@`$\bullet^*$.

The intersection of the patterns is computed against the image $g(h^{-1}(w))$ where $g \subseteq (\Sigma \cup R)^* \times (\Sigma \cup R)^*$ is a regular relation defined by $g(\epsilon)=\epsilon$, $g(\bullet)=\bullet$, $g(r)=r$, $g(f)=\{f, \epsilon\}$, $g(xy)=g(x)g(y)$, for $r \in R$, $p \in P$, $x, y \in (\Sigma \cup R)^*$.

The language C' is a *valid contraction* of the context language C if

$$C \cap h^{-1}(w) = g^{-1}(C' \cap g(h^{-1}(w))) \cap h^{-1}(w). \quad (7)$$

For CG rules, the valid contractions of context languages can be constructed easily because they do not need to express negations through the absence of features.

Compressed Sentence Automaton The language $g(h^{-1}(w)))$ does not give rise to particularly useful representations of the sentence. Instead, a good representation for the substrings of w is obtained by an n-state sentence automaton where each state contains the loops for the feature-value symbols and are connected with the cohort boundary symbols (Figure 5). Under the intersection $C' \cap g(h^{-1}(w)))$, this is a lossless compression because C' still respects the order in P.

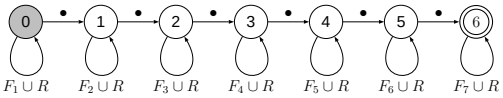


Figure 5: A compressed sentence automaton.

5.4 Reducing the Effect of Rules

The contracted context languages have, rulewise, quite small minimal deterministic recognizers. For example, the rules 343 and 865 give rise to the recognizers in Figure 6.

There are 1377 rules. The sum of the sizes of the recognizers is 10529 states and 16707 arcs. This means, on average, 7.6 states and 12.1 transitions per rule.

A *grammar automaton* recognizes the union of all contracted context languages. For FINCG,

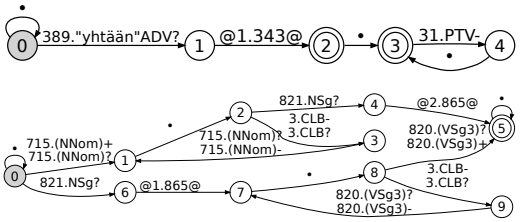


Figure 6: Two contracted context automata.

the obtained minimal deterministic grammar automaton has only 5277 states and 13445 transitions. This saves some space because of the shared states. On average, each rule corresponds to 3.8 states and 9.8 transitions.

A striking thing in this automaton is that 98% of the states have 1 - 7 transitions and three states have 887 - 919 transitions. This suggests that, in the product construction, the computation of the product can be optimized by taking advantage of the size asymmetry inside the product states and the probability of the transitions. If $|F|$ is fixed, the product is computed in $O(ne)$ time where e is the number of states in the grammar automaton.

The total complexity of iterated context testing is bounded by $O(n^2|R|e)$. It is conjectured, however, that the average time complexity of a proper implementation is close to $O(ne \log |R|)$ because (i) the amortized $O(1)$ proximity and refreshing can eliminate the effect of $O(n)$ iterations and (ii) an average cohort is refined by partitioning it into two halves ($\log |R|$ rather than $|R|$).

5.5 Compilation of Contracted Contexts

The Infiltration In formal language theory (Sakarovitch, 2009), the *infiltration* of words $u, v \in \Gamma^*$, denoted with $u \uparrow v$, is defined as the set of words w s.t. subwords u and v cover w completely. More formally, $u \uparrow v$ consists of strings $x_1...x_n \in \Gamma^*$ for which

$$I = \{i_1, ..., i_n\}, i_1 < i_2 < ... < i_n, u = x_{i_1}...x_{i_n},$$
$$J = \{j_1, ..., j_n\}, j_1 < j_2 < ... < j_n, v = x_{j_1}...x_{j_n},$$
$$I \cup J = \{1, ..., n\}.$$

The set $\{\texttt{ABC}, \texttt{ABBC}, \texttt{ABCB}, \texttt{BABC}, \texttt{BACB}, \texttt{BCAB}\} \subseteq \Gamma^*$, for example, is given as $\texttt{AB} \uparrow \texttt{BC}$. The operation extends additively to languages $U, V \subseteq \Gamma^*$ by the definition $U \uparrow V = \{w \in u \uparrow v \,|\, u \in U, v \in V\}$. The infiltration operation is implemented by a state pair construction over automata.

The Synchronization The CG compiler combines conjunctive conditions through the *synchronized infiltration operation* $U \uparrow_{S,<} V$ where $U, V \subseteq \Sigma^*$. The infiltration is restricted in two ways: First, we define a set of *synchronization symbols* S that include the cohort boundary symbol (•), and require that $\{i \mid x_i \in S\} \subseteq I \cap J$ holds for this set. Second, we require that the adjacent letters x_i, x_{i+1} are in a strictly increasing order ($<$) if neither is the cohort boundary symbol (•). Feature symbols $\texttt{SG}+$, $\texttt{SG}-$ and $\texttt{SG}?$, for example, are incomparable letters of Σ, which means that they cannot be adjacent with each other.

The Anchors The relative conditions are anchored to the target cohort by a rule marker. Therefore, rule markers $\texttt{@1.}r\texttt{@}, \texttt{@2.}r\texttt{@} \in R$ are called *anchors*. The linked conditions use other anchors whose shape is $\texttt{@LINK.}n\texttt{@}$ ($n = 1, 2, ...$). Under the synchronized infiltration, the set of synchronization symbols includes the anchors shared by both U and V. The internal anchors are suppressed after they have been used as synchronization symbols. The application of synchronized infiltration to linked conditions is illustrated in Figure 7.

6 The Implementation

The current rule compiler has been created by adapting the skeleton of the Foma tool (Hulden, 2009) to the purposes. The extensions include the infiltration operation, the symbol tables and other essential compiler logic. The compiler is integrated into the parser, whose data structures and algorithms were written from the scratch in order to optimize the computations on cohort automata. Some parts of the system are still under construction. Therefore, experiments on complete parsing are not yet available.

The contracted grammar automaton of 1380 rules is constructed in 20 seconds by the rule compiler. In the parser, about 110 000 cohort automata were read, minimized and mapped to the integer vectors of 1216 features in 1 second (2.2-Ghz Core-2-Duo laptop).

7 Evaluation

The presented parser design has many advantages that concern the space and time requirements and possible extensions. On the other hand, having two finite-state representations of the sentence is

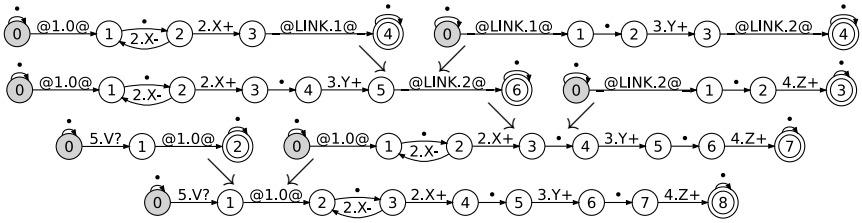


Figure 7: The contracted context for [SELECT (V) ((*1C X) (LINK 1C Y) (LINK 1C Z))]$_0$ is computed with synchronized infiltration and by removing the intermediate link anchors.

clearly a conceptual complication, requiring a special parsing algorithm that ties the representations to each other.

Space Advantages The automaton representation of the grammar is *close to the original grammar size*, fitting easily into cache memories. The product of the grammar automaton and the sentence is *small* and easy to compute.

Speed Advantages The current design is a step towards high-speed CG parsing. This is argued by the following points: (i) The *low space complexity* sets better lower bounds for the time complexity. (ii) The iterations take advantage of the *prior contextual tests*. (iii) The *cohort automata* compress ambiguity. (iv) The *contractions* optimize the constructions. (v) The rules *share* various common parts, which makes the parallel testing of contexts faster. (vi) The combined contextual tests are *compiled* for each rule.

Possible Extensions The current design increases the flexibility of the CG rules: (i) The *application mode* can be altered easily. (ii) *Coordinated disambiguation rules* can be implemented for syntagmatic patterns. (iii) The context conditions can be arranged to *levels of exceptions* (@1.r@, @2.r@, @3.r@,...). (iv) The rule *formalism* can be extended. (v) The borderline between morphological and syntactic rules can be removed by *lexicalizing* the intermediate mapping.

8 Conclusions

The paper has described a nonconventional CG parser architecture using finite-state methods. In the approach, the list representation of the readings is replaced with a cohort automaton and feature vectors. The readings and contexts are tested with contracted patterns. Iterated testing is optimized with inward processing. The presented architecture and its prototype are expected to lead to an efficient and flexible parser and enrich the related research.

Acknowledgments

The work has been supported by the grant #128536 “Open and Language Independent Automata-Based Resource Production Methods for Common Language Research Infrastructure” of the Academy of Finland. I am also indebted to A. Voutilainen, F. Karlsson, K. Koskenniemi, K. Lindén, and T. Trosterud for long-time interest, PSC 2010 reviewers for suggestions, CG 2011 participants for helpful responses in Riga, and for M. Hulden for discussions in Blois.

References

Steven Abney. 1991. Parsing by chunks. In Robert Berwick, Steven Abney, and Carol Tenny, editors, *Principle-Based Parsing*. Kluwer Academic Publishers.

Salah Aït-Mokhtar and Jean-Pierre Chanod. 1997. Incremental finite-state parsing. In *Proc. 5th ANLP, the Conference on Applied Natural Language Processing*, pages 72–79, Washington, DC.

G. Edward Barton, Jr. 1986. Constraint propagation in Kimmo systems. In *Proc. 24th ACL*, pages 42–52, New York, NY, July 10-13. The Association for Computational Linguistics (ACL), Stroudsburg, PA.

Glenn David Blank. 1989. A finite and real-time processor for natural language. *Communications of the ACM*, 32(10):1174–1189, October.

Henning Bordihn, Henning Fernau, Markus Holzer, Vincenzo Manca, and Carlos Martín-Vide. 2006. Iterated sequential transducers as language generating devices. *Theoretical Computer Science*, 369:67–81.

Eric Brill. 1992. A simple rule-based part of speech tagger. In *Proc. 3rd ANLP*, Trento, Italy

Kenneth Ward Church. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *Proc. 2nd ANLP*, pages 136–143, Austin, TX.

Tino Didriksen. 2010. Constraint Grammar Manual: 3rd version of the CG formalism variant. GrammarSoft Aps, Denmark.

Jorge Graña, Gloria Andrade, and Jesús Vilares. 2003. Compilation of constraint-based contextual rules for part-of-speech tagging into finite state transducers. In *Proc. 7th CIAA, the Conference on Implementation and Application of Automata*, pages 128–137, Springer-Verlag, Berlin, Germany.

Gregory Grefenstette. 1999. Light parsing as finite state filtering. In András Kornai, editor, *Extended finite state models of language*, pages 86–94. Cambridge University Press, New York, NY.

Maurice Gross. 1968. *Grammaire transformationnelle du francais*, volume 1, Syntaxe du verbe. Larousse, Paris, France.

David A. Huffman. 1971. Impossible Objects as Nonsense Sentences. *Machine Intelligence*, 6:295–323.

Mans Hulden. 2009. Foma: a finite-state compiler and library. In *Proc. Demonstrations Session at the 12th EACL*, pages 29–32, Athens, Greece. ACL, Stroudsburg, PA.

Mans Hulden. 2011. Constraint grammar parsing with left and right sequential finite transducers. To appear in *Proc. 9th FSMNLP*, Blois, France. ACL, Stroudsburg, PA.

C. Douglas Johnson. 1972. *Formal Aspects of Phonological Description*. Number 3 in Monographs on linguistic analysis. Mouton, The Hague.

Aravind K. Joshi and Phil Hopely. 1996. A parser from antiquity. *Natural Language Engineering*, 2(2):291–294.

Aravind K. Joshi and Leon S. Levy. 1982. Phrase structure trees bear more fruit than you would have thought. *American Journal of Computational Linguistics*, 8(1):1–11.

Tomasz Jurdziński, Friedrich Otto, František Mráz, and Martin Plátek. 2005. Deterministic two-way restarting automata and Marcus contextual grammars. *Fundamenta Informaticae*, 64(1–4):217–228.

Fred Karlsson, Atro Voutilainen, Juha Heikkiä, and Arto Anttila, editors. 1995. *Constraint Grammar: a Language-Independent System for Parsing Unrestricted Text*, volume 4 of *Natural Language Processing*. Mouton de Gruyter, Berlin and New York.

Fred Karlsson. 1990. Constraint Grammar as a framework for parsing unrestricted text. In H. Karlgren, editor, *Proc. 13th COLING*, volume 3, pages 168–173, Helsinki, Finland. International Committee on Computational Linguistics (ICCL).

Lauri Karttunen. 1997. The replace operator. In Emmanuel Roche and Yves Schabes, editors, *Finite-State Language Processing*, chapter 4, pages 117–147. A Bradford Book, the MIT Press, Cambridge, MA, USA.

Lauri Karttunen. 1998. The proper treatment of optimality in computational phonology. In *Proc. 2nd FSMNLP*, pages 1-12, Bilkent University, Ankara, Turkey.

Kimmo Koskenniemi, Pasi Tapanainen, and Atro Voutilainen. 1992. Compiling and using finite-state syntactic rules. In *Proc. 14th COLING*, volume I, pages 156–162, Nantes, France. International Committee on Computational Linguistics (ICCL).

Kimmo Koskenniemi. 1983. *Two-level morphology: a general computational model for word-form recognition and production.* Ph.D. thesis, number 11 in Publications of the Department of General Linguistics, University of Helsinki. Yliopistopaino, Helsinki, Finland.

Kimmo Koskenniemi. 1997. Representations and finite-state components in natural language. In Emmanuel Roche and Yves Schabes, editors, *Finite-state language processing*, chapter 3, pages 99–116. A Bradford Book, The MIT Press, Cambridge, MA.

Steven Krauwer and Louis des Tombe. 1981. Transducers and grammars as theories of language. *Theoretical Linguistics*, 8:173–202.

Torbjörn Lager and Joakim Nivre. 2001. Part of speech tagging from a logical point of view. In P. de Groote, G. Morrill, and C. Retoré, editors, *Logical Aspects of Computational Linguistics*, volume 2099 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 212–227. Springer-Verlag, Berlin, Germany.

Torbjörn Lager. 2001. Transformation-based learning of rules for constraint grammar tagging. Presented at the 13th Nordic Conference in Computational Linguistics, NODALIDA, Uppsala, Sweden, May 21-22.

Nikolaj Lindberg and Martin Eineborg. 1998. Learning constraint grammar-style disambiguation rules using inductive logic programming. In *Proc. 36th ACL / 17th COLING, Montréal, Quebec, Canada, August 10-14*, volume 2, pages 775–779. ACL, Stroudsburg, PA.

Mitchell P. Marcus. 1980. *A Theory of Syntactic Recognition for Natural Language*. The Series in Artificial Intelligence. The MIT Press, Cambridge, MA.

Hiroshi Maruyama. 1990. Structural disambiguation with constraint propagation. In *Proc. 28th ACL*, pages 31–38, Pittsburgh, PA. ACL, Stroudsburg, PA.

Denis Maurel. 1989. *Reconaissance de sé quences de mots par automates. Adverbes de date*. Ph.D. thesis, Université Paris 7, Paris, France.

Mehryar Mohri. 1994. Syntactic analysis by local grammars automata: an efficient algorithm. In D. K. Kiefer, G. Kiss, and J. Pajzs, editors, *Proc. International Conference on Computational Lexicography (COMPLEX 94)*, pages 179–191, Budapest, Hungary.

Kemal Oflazer and Gökhan Tür. 1997. Morphological disambiguation by voting constraints. In *Proc. 35th ACL / 8th EACL*, pages 222–229, Madrid, Spain. ACL, Stroudsburg, PA.

Ivan Petrov Peikov. 2006. Direct construction of a bimachine for context-sensitive rewrite rule. Master's thesis, Sofia University St. Kliment Ohridski, Faculty of Mathematics and Computer Science, Department of Mathematical Logic and Applications, Sofia.

Janne Peltonen. 2011. Rajoitekielioppien toteutuksesta äärellistilaisin menetelmin. Master's thesis, University of Helsinki, Department of Modern Languages, Helsinki.

Paul Stanley Peters and Robert W. Ritchie. 1969. Context sensitive immediate constituent analysis — context-free languages revisited. In *Proc. ACM Symposium on Theory of Computing*, pages 1–8, Marina del Rey, California, May 5–7.

Martin Plátek, Markéta Lopatková, and Karel Oliva. 2003. Restarting automata: motivations and applications. In M. Holzer, editor, *Workshop Petrinetze und 13. Theorietag Automaten und Formale Sprachen*, pages 90–96, Institut für Informatik, Technische Universität München, München, Germany.

Emmanuel Roche. 1994. Two parsing algorithms by means of finite state transducers. In *Proc. 20th COLING*, volume 1, pages 431–435, Kyoto, Japan. International Committee on Computational Linguistics (ICCL).

Emmanuel Roche. 1997a. Compact factorization of finite-state transducers and finite-state automata. *Nordic Journal of Computing*, 4(2):187–216.

Emmanuel Roche. 1997b. Parsing with finite-state transducers. In Emmanuel Roche and Yves Schabes, editors, *Finite-state language processing*, chapter 8, pages 241–281. A Bradford Book, the MIT Press, Cambridge, MA.

Jacques Sakarovitch. 2009. *Elements of Automata Theory*. Cambridge University Press, Cambridge, NY.

Nicolae Santean and Sheng Yu. 2006. On weakly ambiguous finite transducers. In O.H. Ibarra and Z. Dang, editors, *DLT 2006*, volume 4036 of *LNCS*, pages 156–167. Springer-Verlag, Berlin, Germany.

Wojciech Skut, Stefan Ulrich, and Kathrine Hammervold. 2004. A bimachine compiler for ranked tagging rules. In *Proc. 20th COLING*, Geneva, Switzerland. International Committee on Computational Linguistics (ICCL).

Pasi Tapanainen. 1996. *The Constraint Grammar Parser CG-2*. Number 27 in Publications of the Department of General Linguistics, University of Helsinki. Yliopistopaino, Helsinki, Finland.

Pasi Tapanainen. 1999. *Parsing in two frameworks: finite-state and functional dependency grammar*. Ph.D. thesis. Department of General Linguistics, University of Helsinki, Finland.

Atro Voutilainen. 1994. *Three studies of grammar-based surface parsing of unrestricted English text*. Ph.D. thesis, number 24 in Publications of the Department of General Linguistics, University of Helsinki. Yliopistopaino, Helsinki, Finland.

William A. Woods. 1970. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10):71–87. Association for Computing Machinery (ACM).

Anssi Yli-Jyrä. 1995. Schematic finite-state intersection parsing. In Kimmo Koskenniemi, editor, *Short Papers Presented at the 10th Nordic Conference of Computational Linguistics (NODALIDA-95)*, pages 95–103, Helsinki, Finland, 29–30 May.

Anssi Yli-Jyrä. 1997. Menetelmiä äärellisiin automaatteihin perustuvan lauseenjäsennyksen tehostamiseksi. Master's thesis, Department of General Linguistics, University of Helsinki, Helsinki, Finland.

Anssi Yli-Jyrä. 2005. *Contributions to the Theory of Finite-State Based Grammars*. Ph.D. thesis, number 38 in Publications of the Department of General Linguistics, University of Helsinki. Yliopistopaino, Helsinki, Finland.

Anssi Yli-Jyrä. 2007. Transducers from parallel replacement rules and modes with generalized lenient composition. In Thomas Hanneforth and Kay-Michael Würzner, editors, *Finite-State Methods and Natural Language Processing, 6th FSMNLP, Revised Papers*, pages 197–212, Potsdam University Press, Potsdam, Germany.

Anssi Yli-Jyrä. 2010. Efficient context-sensitive rewriting with inward deterministic transducers. A submitted manuscript.

Anssi Yli-Jyrä. 2011a. Compiling simple context restrictions with nondeterministic automata. To appear in *Proc. 9th FSMNLP*, Blois, France. ACL, Stroudsburg, PA.

Anssi Yli-Jyrä. 2011b. Explorations on positionwise flag diacritics in finite-state morphology. In Bolette Sandford Pedersen, Gunta Nešpore, and Inguna Skadin, editors, *NODALIDA 2011 Conference Proceedings*, pages 262–269, Riga, Latvia.

An Experiment of Use and Reuse of Verb Valency in Morphosyntactic Disambiguation and Machine Translation for Euskara and North Sámi

Linda Wiechetek
Giellatekno / Romssa universitehta
linda.wiechetek@uit.no

Jose Mari Arriola
IXA / Euskal Herriko Unibertsitatea
josemaria.arriola@ehu.es

1 Introduction

There are a number of well known resources dealing with verb valency including *PropBank* (Palmer et al., 2005), *VerbNet* (Kipper et al., 2006) and *VALLEX* (Hajič et al., 2003). These include thematic roles, morpho-syntactic specifcations and selection preferences. A comparatively wide definition of valency including subcategorization information on all mentioned linguistic levels is applied here. However, these resources have not often been used in rule-based NLP tasks such as machine translation or disambiguation. 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 other elaborated linguistic resources. Bick (2000) uses tags specifying transitivity preferences such as "preferably transitive, but potentially intransitive" but also selection preferences, e.g. specifying a human accusative. Agirre et al. (2009) successfully apply valency information, i.e. case subcategorization information, to the Spanish->Euskara MT system *Matxin* in order to improve NP/PP translation. They present different kinds of tests enriching their machine translation system with different techniques. In all cases, the combinations of techniques that include valency information produce the best results especially in recall and F-score.

This paper describes an experiment for the application of verb valency in Euskara and North Sámi rule-based NLP applications, i.e. morpho-syntactic disambiguation and machine translation. 10 frequent verbs each are annotated to improve the analysis, and later the effects on the application is evaluated.

The main objective of the experiment is improving linguistic resources for North Sámi and Euskara taking advantage of pre-existing existing resources in one language and transfering them to the other language. Other works (Antonsen et al., 2010) have shown that the reuse of grammatical resources between both related and unrelated endangered languages is possible and provides useful results, especially on a high level of linguistic analysis. In Antonsen et al. (2010) especially the reuse of the dependency grammar is described.

A number of problems that syntax alone cannot handle can be resolved by semantically richer information included in verb valency. Verb valency annotation is applied on a high level of linguistic analysis and is therefore useful for reuse even for unrelated languages.

2 The experiment

The test cases used in this experiment regard morpho-syntactic disambiguation and machine translation and improve the analysis / translation by making use of valency information.

In many cases, pure syntactic information is not sufficient for the morpho-syntactic disambiguation of nouns, and richer linguistic information is needed. The same counts for machine translation, where morpho-syntactic generation of nouns and polysemy resolution can require high-level linguistic analysis.

The languages in question are lesser-used languages, with 15,000 to 25,000 North Sámi speakers and 775,000 Euskara speakers. North Sámi and Euskara are unrelated languages. Euskara is a language isolate, while North Sámi belongs to the Finno-Ugric language family. One major similarity is their morphological complexity: Euskara is an agglutinative language and North Sámi has both agglutinative and inflective features. They also both have a medium to large sized system of affixed case markers/postpositions. North Sámi has 7 cases (nominative, genitive, accusative, locative, illative, comitative, essive), while Euskara has 17 affixed cases/postpositions (ergative, absolutive, possessive genitive, local genitive, dative, allative, ablative, inessive, destinative, partitive, prolative, instrumental, sociative, motivative, directional and terminative) [1]. In North Sámi, two of the main ambiguities are genitive-accusative and comitative-locative with a significant impact on the F-Score of the analysis.

In Euskara, the homonymy of absolutive plural and ergative singular cause approximately 40% of the ambiguity left after morpho-syntactic disambiguation.

[1] The definition of the terms case/postposition is disputed. In the current terminology only ergative, absolutive and dative are considered cases, while the others are considered affixed postpositions.

(1) Nekez lortu nuen zure ezpainak ikustea.
Hardly achieve do-I-IT.PAST your lip-ERG.SG/ABS.PL see-VNOUN.ABS.SG
'I hardly managed to see your lips.'

In example (1), 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 the object reading holds, therefore absolutive case should be selected. The ergative interpretation can be discarded based on valency requirements of the nominalized verb *ikustea 'seeing'*. The ambiguity is not resolved in the current version of the Euskara analyzer, but can be resolved by similar methods as in North Sámi.

2.1 Technical and linguistic background

The Euskara and Sámi sentence analysis NLP tools built at the University of the Basque country and the University of Tromsø have a similar structure. They contain finite-state transducers for the morphological analysis compiled with the Xerox compilers `twolc` and `lexc` (Beesley and Karttunen, 2003). They can alternatively be compiled with the open-source compilers HFST (Lindén et al., 2009) (North Sámi) and foma (Alegria et al., 2010) (Euskara). For syntactic analysis and morpho-syntactic disambiguation, Constraint Grammar parsers are being used (Karlsson, 2006).

One main difference between the systems is that while for North Sámi, morphological and syntactic modules are strictly separated, for Euskara most of the syntactic tags are annotated in the same module that adds morphological information to the lemmata. The thought behind that was that morphology and syntax are closely related and in a number of cases the syntactic function can be unambiguously mapped to the morphological representation.[2] In North Sámi mapping the tags at the same time would lead to imense overgeneration, as there is a huge amount of homonymy. The syntactic mapping and disambiguation is organized a bit differently / takes different philosophies as their basis. North Sámi has a large mapping section where by means of context specifications, secure syntactic tags are mapped and disambiguated at an early phase. In Euskara, most of the syntax is first introduced with all ambiguity regardless of the context, and later select and remove rules take care of disambiguation. Another difference is that Euskara has several seperate modules that treat different syntactic tasks, in North Sámi, one grammar handles all of the syntactic tag mapping except for explicit dependencies. Some of the modules that are used for Euskara, mainly the chunking module, which is introduced to handle dependencies, do not exist for North Sámi. In North Sámi, recognizing chunks (such as relative clauses etc.) is done implicitly by selecting barriers for phrases and making classes of clause-boundary identifiers.

North Sámi machine translation uses the open-source rule-based machine translation platform Apertium[3] (Forcada et al., 2009). There are existing prototypes for North Sámi Lule Sámi (sme-smj), South Sámi (sme-sma), Finnish (sme-fin), Norwegian (sme-nob) and Euskara (eus-sme). Apertium works with shallow transfer and uses finite-state transducers, hidden Markov models (HMM), Constraint Grammar and finite-state based chunking.

Most of the ambiguity for North Sámi can be resolved by means of sets for verb valency, semantic prototype sets for nouns and linguistic rules that make use of those (Trosterud and Wiechetek, 2007). There are approximately 60 sets that categorize the verbs according to the syntactic cases they subcategorize for and approximately 160 sets of nouns according to their semantic properties. CG morpho-syntactic rules apply this information and rule out either one of the cases (Trosterud and Wiechetek, 2007).

A set specifying the syntactic subcategorization is for example **LOCV** containing verbs like *ballat 'fear'* and *jearrat 'ask'*. It is used in a rule asking for an argument in locative case. A set specifying the semantic subcategorization on the other hand is **PLACE-V** containing verbs such as *čuožžut 'stand'* and *orrut 'live'*. It is used in a rule typically selecting locative instead of comitative case if the argument is a noun denoting a place.

For Euskara, a few general semantic features derived from the machine translation system MATXIN (Mayor et al., 2011) such as **ANIMATE, HUMAN, TIME, MATERIAL, VEHICLE** and **LANGUAGES** are used. North Sámi on the other hand has also more specific sets, such as **EDUCATION** containing words like *skuvla 'school' giellagursa 'language class'* and **PLACE** containing words like *jeaggi 'swamp'*, *luossabáiki 'salmon fishing place'* and *gávpot 'city'*.

For complex NLP tasks, often a systematized way of storing valency information is desirable. Subcategorization information of verbs (and other PoS as well) is more complex than simple semantic categorization of nouns as it includes morphological, syntactic and semantic information, which is related not only to the verb itself but to a number of arguments that are potentially related to the verbs. Multiple dimensions need to be considered when working with valency.

Sets to encode subcategorization information for verbs encode the information in a fairly one-dimensional way. The main disadvantage of the codification of valency information in sets is that

[2] The ergative plural suffix *-ek* is always a subject *@SUBJ*.

[3] `http://www.apertium.org`

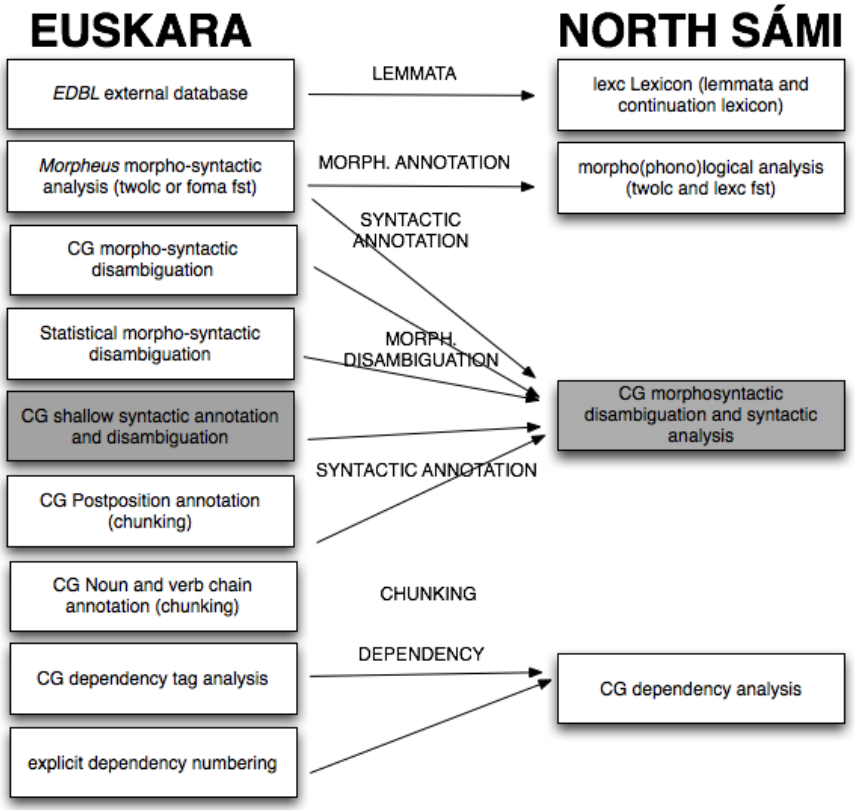


Figure 1: Comparison of the chains for Euskara and North Sámi; the shaded areas mark the places where valency annotation is included

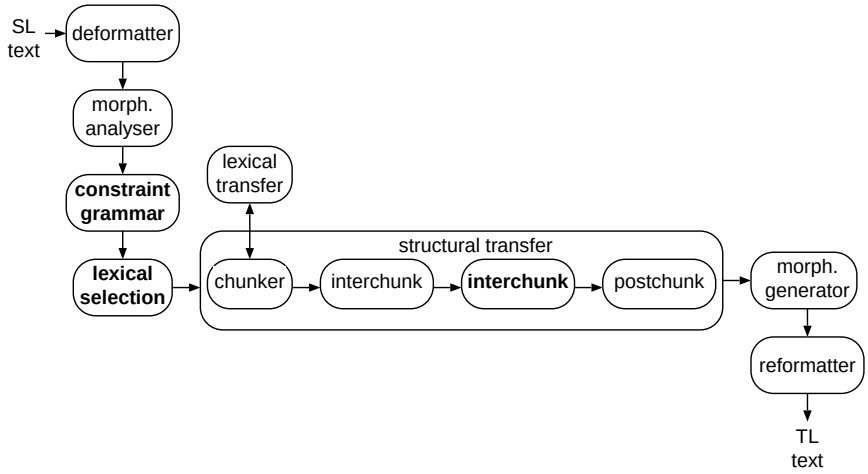


Figure 2: The chain of modules in Apertium

```
<verb lm="hil">
  <frame id="1">
    <ex>Miren hil da.</ex>
    <glosses>
      <gloss lang="eng">Miren has died.</gloss>
      <gloss lang="sme">Miren lea jápmán.</gloss>
    </glosses>
    <theme>
      <case>abs</case>
      <syn>subj</syn>
      <sem>animate</sem>
    </theme>
  </frame>
  ...
```

Figure 3: Verb valency information

the information cannot always be accessed as a whole. In some cases the lemmata in the sets are polysemous, and only one of the meanings is relevant in a certain context. For example, a rule applying the previously named set of place-verbs **PLACE-V** hits for the Sámi verb *orrut '1. stay, live 2. seem*, where it should only hit when the first sense *'stay, live'* is used. If a full valency specification of the verb is available, this problem can be avoided e.g. in tasks as MT as the word senses can be distinguished together with their valencies. Therefore a multi-dimensional representation of valency information is desired.

For Euskara on the other hand, an elaborated database of 100 verbs originally developed for the Euskara PropBank Aldezabal et al. (2010) already exists. It contains rich valency information, i.e. semantic frames including semantic roles and morpho-syntactic information. Here the valency aspect is approached very much from the linguistic side and not so much based on NLP problems (e.g. ambiguity, lexical selection etc.). The challenge of applying the database to NLP tasks lies in the adaptions that are necessary in order to resolve NLP specific problems. As Bick (2000) claims, his categories are made to distinguish meaning, not to define it.

2.2 Annotation of frames

In the experiment, (Aldezabal et al., 2010) verb valency information is converted into valency tags for both Euskara and North Sámi that can be used for disambiguation and machine translation. Code 3 illustrates the way the information is encoded in the database that is meant to be used at a later stage of development.

Each verb can have several frames of argument constellations. The Euskara verb *hil 'die; kill'* for example has two frames, one for the sense *die* and the other for the sense *kill*. In the first sense it has only one argument, in the second it has two.

```
1 hil V Thcase\_Abs Thsyn\_Subj Thsem\_Ani
2 hil V Agcase\_Erg Agsyn\_Subj Pacase\_Abs
        Pasyn\_Obj Pasem\_Ani
```

Arguments are ordered by semantic roles (e.g. agent, theme, topic, patient, location) because they are more unique[4] 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 *hil 'die; kill'*, the first argument, characterized by the semantic role theme, has the three attributes **Thcase_Abs** (absolutive case) **Thsyn_Subj** (syntactic function subject) **Thsem_Ani** (selection restriction animate).

For each verb in Euskara, each frame had to be matched to a verb in North Sámi. In many cases, a lemma in Euskara could not be directly translated to a Sámi verb, the valency frames had to be taken into account to find the correct equivalent(s). When the equivalent in Sámi had been found, the frames were copied to Sámi, and in a second step adaptions were made. While roles in principle stayed the same, cases and to a smaller degree also syntactic functions had to be changed.

For test purposes, we found that the easiest way to annotate valency information was by means of Constraint Grammar rules.

The rules adding the valency tags to the verb *lortu 'achieve, get'* have the following format:

```
ADD (Agcase_Erg Agsyn_Subj 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 to include the tags automatically from the verb database in figure 3.

2.3 Disambiguation

Verb valency information is used for syntactic disambiguation. In Euskara, both morphological and syntactic ambiguity exist, i.e. one word recives multiple analyses. Morphological ambiguity in Euskara includes e.g. categorial ambiguity e.g. typically noun/verb ambiguity. For agglutinative languages there are additional sources of ambiguity (number, case, etc.). One of the most pervasive ambiguities is the one related to the suffix *–ak*, it can codify absolutive plural or ergative singular. Additionally the suffix *-a* causes significant ambiguity.

Syntactic ambiguity is added on top of morphological ambiguity. Disambiguation of subject or object functions is needed to detect agreement errors. Concerning the previously mentioned suffix *–ak* the following ambiguity is given: absolutive case can be subject, object or predicative, ergative case on the other hand can only be a subject.

[4]Constellations with e.g. two themes are possible but not that frequent.

This suffix can be attached mainly to nouns and also finite verbs (e.g. *etorri den-ak 'the one who has come'*) or non-finite verbs (e.g. *etortze-ak 'the coming'*), which are converted into subordinate clauses. Here, the ambiguity appears in much more complex contests: finite or non-finite verbs with subject, object or predicative function. The same ambiguity is caused by the suffix *–a*.

In order to improve morpho-syntactic disambiguation, valency information is used to reduce absolutive/ergative ambiguity and syntactic ambiguity. Both morpho-syntactic and syntactic ambiguity are closely related. For that reason, during the first step absolutive or ergative case is selected, and in the second step the correct syntactic tag is chosen. This is done in a second constraint grammar module. This 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 BARRIER
  ADI/ADL/ADT LINK 0 Thsyn_Obj );
```

The other ambiguity in the sentence consists in the readings of the 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-FINITE-VERB_CLAUSE_OBJ)
  IF (0 NON-FINITE-VERB)
  (*-1 Thsyn_Obj BARRIER ADI/ADL/ADT);
```

Even though semantic roles are not explicitly annotated to the verbs' arguments in running text, the CG-rules make use of semantic role information as in *Thsyn_ Obj 'the theme has the syntactic function object'*. In order to annotate semantic roles, Bick and Valverde (2009) uses morphological information (PoS, case etc.), syntactic information (subj etc) and semantic information. In addition, barriers that identify beginning and end of a phrase, are necessary to define the dependency between verbs and their arguments, especially to find long-distance dependencies in case there are relative (subordinate) phrases etc. In general, it can be said that by means of two elements of the "triple" valency, semantic roles and dependencies, the third one can be identified.

Therefore, it makes sense to refer to semantic roles of the arguments of the verbs, even at this stage of the analysis. Furthermore, semantic roles are currently being annotated in the corpus of Euskara, and will be available in the near future.

2.4 Machine Translation

In the case of machine translation, the use of valency is meaningful for two subtasks. In the first case, a default argument realization is corrected to the one that suits the valency requirements/restrictions in the target language. Sociative case in Euskara usually corresponds to comitative case in Sámi (cf. Table 1).

Euskara	**North Sámi**
ergative	nominative
absolutive	accusative, (nominative, essive)
genitive	genitive
inessive	locative
ablative	locative
dative	illative
allative	illative
benefactive	illative
instrumental	comitative
sociative	comitative

Table 1: Default correspondences between a number of relevant cases in Euskara and North Sámi

In some cases, the verb valency in the target language deviates from the default case correspondence as in example (2-a), where the case of the experiencer is illative as assigned by a substitution rule.

(2) a. Zergatik haserretzen zara nirekin?
Why get.angry do.you I.soc.sg

'Why do you get angry with me'

b. Manin don suhtat munnje?
Why you get.angry I.ill.sg
'Why do you get angry with me'

The following substitution rules assign valency within a separate Apertium valency module to the verbs in Euskara and North Sámi.

```
SUBSTITUTE (V) (V Caucase_Erg Caucase_Soc
  Causyn_Subj Expcase_Abs Expsyn_Obj
  Expsem_Hum) ("haserre");

SUBSTITUTE (V) (V Caucase_Nom Causyn_Subj
  Causem_Hum Expcase_Ill Expcase_ala
  Expsyn_Advl Expsem_Hum) ("suhttat");
```

Another substitute rule in a constraint grammar valency module replaces the Euskara valency frame for *haserre* by the North Sámi valency frame and a transfer rule matches the correct case to the Sámi noun based on the case attributes in the valency frame.

As a default, a transfer rule as shown in 5 selects a the most frequent corresponding case, e.g. comitative, for a particular case, here sociative, in Euskara.

The following rule picks a valency-based case, if a verb valency tag asks for it. It sets case to +Acc if Thcase is Thcase_Acc.

```
SUBSTITUTE (%Val Cacase_Erg Cacase_Soc
    Casyn_Subj Excase_Abs Exsyn_Obj Exsem_Hum)
    (%Val Cacase_Ill Cacase_ala Casyn_Advl
    Casem_Hum Excase_Nom Exsyn_Subj Exsem_Hum)
    ("haserre");
```

Figure 4: Substitution rule in the valency module

```
<choose>
  <when>
      <test><equal><clip pos="1"
         side="sl" part="case"/>
         <lit-tag v="Soc"/></equal></test>
  <let><clip pos="1" side="tl" part="case"/>
         <lit-tag v="Com"/></let>
  </when>
</choose>
```

Figure 5: Transfer of default cases

In the other case, i.e. lexical selection, depending on the valency frame of the verb, a specific lexeme is chosen in the target language. The regular case when translating from one language to the other, i.e. from Euskara to North Sámi, is that there is more than one possible translation depending on the context, i.e. in most cases the valency frame of the verb. The verb *hil 'die, kill'* translates into *jápmit 'die'* with only the theme role realized. With an animate patient object, it translates into *goddit 'kill'*.

The lexical selection module helps to pick the correct equivalent. The lexicon specifies the possible lexical variants by means of numbers.

```
hil jápmit (die)
hil:1 goddit (kill)
```

A lexical selection rule picks the non-default reading *goddit 'kill'* if it finds an animate absolutive item to the left of it.

```
SUBSTITUTE ("hil") ("hil:1") ("hil")
  (0 (Pacase_Abs Pasyn_Obj Pasem_Ani)
  LINK *§PA LINK 0 ANIMATE BARRIER FAUX
  OR S-BOUNDARY2);
```

3 Evaluation

3.1 Translation of frames

100 Euskara verbs were translated into 187 North Sámi verbs on a frame-to-frame basis, i.e. a polysemy of at least 1,87 meanings per Euskara verb as can be seen in table 2. Careful lexicography work would of course increase the number of possibilities. Of the 187 translations some doubles were found, e.g. *mannat* 'go' (6x), *boahtit* 'come' (5x), *leat* 'be' (3x), *borrat* 'eat' (4x), *šaddat* 'become' (3x). The 100 verbs have 219 listed frames, 184 of those correspond to frames of North Sámi verbs, 35 do not. The correspondence is based on semantic roles, not on syntactic correspondence or on case correspondence.

Of the 35 that do not correspond, there are different types: some of the verbs lexicalize different parts of the argument structure. While in Euskara, *barkatu 'forgive'* and *afaldu 'have dinner'* consist of only one lexical unit, in North Sámi part of the verb is realized as an argument *addit ándagassii 'forgive'* and *borrat eahketbiepmu 'have dinner'* and therefore changes the argument structure quantitatively. Verbs that do not correspond, both quantitatively and qualitatively are motion verbs such as *atera 'go out'*, *etorri 'come'*,*igo 'ascend, rise'*, *iritsi 'arrive'*, *pasatu 'go by'*, *sartu 'enter'* *eraman 'bring'*. While in Euskara, typically both source and destination are defined, in North Sámi only either one belongs to the argument scheme.

	Euskara	**N. Sámi**
verbs	100	187
frames	219	-
- corresponding		184
- not corresponding		35

Table 2: Valency-based polysemy and correspondence between verb frames

Typically, a change in valency also corresponds to another translation equivalent. In some cases, all frames of one verb in the source language are translated with one verb in the target language, as is the case for the verb *elkartu 'meet'*, which translates into *deaivvadit*. In other cases polysemy is not related to a distinction in frames. The verb *jo* in its sense 'hit' for example translates into *časkit* if the agent is a human. If the agent is a e.g. a horse, it translates into *nordadit*. Here semantic selection restrictions are necessary for a lexical selection. But in general, semantic role based valency seems to be very useful for a basic sense distinction and lexical selection in machine translation.

```
<choose>
  <when>
    <test><equal><var n="Thcase"/>
        <lit-tag v="Thcase_Acc"/>
        </equal></test>
    <let><clip pos="1" side="tl" part="case"/>
        <lit-tag v="Acc"/></let>
  </when>
</choose>
```

Figure 6: Transfer: valency-based case selection

3.2 Disambiguation

10 of the most frequent verbs for disambiguation in Euskara were tested and evaluated. The test corpus contains 177 verbs, the verbs evaluated in the experiment represent 5,6% of the verbs of the sample.

The valency frames of 10 verbs were annotated by means of 48 mapping rules. The grammar contains 47 disambiguation rules that resolve absolutive / ergative, absolutive sg./ absolutive indefinite, object, subject and predicative ambiguity for Euskara. The rules can refer to valency tags rather

than the verb lemma and therefore apply to any verb with the characterisitics that appear in the context specification of the rule. The testcorpus is running text and includes sentences without the verbs that have been annotated. This leads to low coverage on the one hand, but takes into account the general impact of the annotation with respect to the frequency of the selected verbs on the other hand. The precision and recall for the rules involved in the disambiguation of the absolutive-ergative syncretism case are 72% and 72% respectively. While the overall analysis improves by 5.5 %, the figures for the ambiguity resolution aimed at are higher. As can be seen in table 3, abs.pl. - erg.sg. ambiguity resolution improves by 18 %, and abs.sg. - abs. indef. ambiguity by 24 %.

precision	72 %
recall	72 %
disambiguation of	
... abs.pl. - erg.sg.	18 %
... abs.sg. - abs. indef.	24 %
... abs./erg. - abs. sg./indef.	46.4 %
improvement for	
... overall analysis	26.9 %
... abs./erg. - abs. sg./indef.	46.4 %

Table 3: Evaluation of morpho-syntactic disambiguation for Euskara

In 13 of 83 cases, the subcategorization information for the verbs is missing completely, in the remaining 19 cases the existing rules do not manage to disambiguate correctly. 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 and low coverage of semantically annotated nouns.

In 13 of those 19 erroneously applied rules, the rule disambiguates based on the valency information of an unrelated verb, in the 6 remaining cases the semantic information of the nouns is missing. In order to improve the results generalising 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.

3.3 Machine Translation

Valency information has been used for two distinctive tasks in machine translation, syntactic transfer (e.g. picking the correct morpho-syntactic realization of the arguments) and lexical selection (picking the correct lexical equivalent in the target language). Since the basic free resources that are necessary for a complete analysis are not available, they cannot be included in the open-source Apertium machine translation system and only the lexical selection rules have been evaluated. 10 verbs have been annotated and 29 rules refering to valency information have been made for lexical selection. Test sentences to evaluate the lexical selection rules are taken from a newspaper corpus of Euskara. This evaluation will be aimed at improving the existing hand-written rules. As such it will be qualitative not quantitative. A full quantitative evaluation is not possible as the non-availability of existing grammatical resources prevents automatic analysis. The evaluation is focussed on explaining why in some cases where the rules do not apply. Usually this is not because it is impossible to formulate a rule for a given context, but rather that a linguist is not able to forsee all possible contexts without real-life sentences and extensive corpus analysis.

The lexical selection rules are built in the following manner: They refer to a possible right and a possible left context with a semantic role often linked to a case or syntactic function. The context is restrained by a barrier taking into consideration possible markers of borders of clauses such as other finite verbs, punctuation and subordinators. It is obvious that these rules could easily be too simple and that their constraints may have to be modified. With a dependency annotation of the the relations in the sentence between the arguments and the verb would be explicit and barriers would not be necessary.

Rules for lexical selection that refer to quantitative valency differences (differentiating between translation equivalents by means of the number of arguments) as in the case of *hil 'die; kill'* seems to be pretty straightforward. The only difference is that one has only a theme, while the other has an agent and patient. In case of a missing agent, the *jápmit `die'* reading could be selected. The difficulty is that in Euskara the agent does not have to be explicit. Both subject, object and indirect object can be dropped. The auxiliary on the other hand is explicit about the number of grammatical arguments, if the agent is missing another form of the auxiliary is being used. But the auxiliary can be missing too, either when it has the form of a nominalization as in *hiltzea '(the) dying/killing'* or when preceeding a postposition as in *29 lagun hil ondoren 'after 29 people had died' or 'after they had killed 29 people'*.

When the decisive differences for lexical selection are qualitative rather than quantiative, e.g. for *asmatu 'guess, invent, think'* which can be translated as *árvidit 'guess'* or *fuomášit 'invent, come up with, think'* subject/object drop can become a problem. If it is translated as *árvidit 'guess'* it has a theme role while *fuomášit 'invent, come up with, think'* can have a product role. Furthermore it needs to be taken into account that semantic roles can also be carried by clauses such as *itua bete betean asmatzen zutenak 'the ones that guessed the aim exactly'*, where the auxiliary *zutenak 'the ones that did'* carries the semantic role. It makes therefore more sense if rules refer to syntactic functions

rather than morphological cases as carrier of semantic roles.

With regard to barriers, it is important to take into account how far the dependencies of a verb span. In some cases the valency spans far (3), in others they do not (4).

(3) - ...zer-nolako harrera egin-go zion asmatzera jarri zen
what-how welcome make-FUT do-PAST.SUBJ.3.SG.OBJ.3.SG.IOBJ.3.SG think bring be-PAST.SUBJ.3.SG
she/he started thinking what kind of welcome he/she would make her

Here a whole clause *zer-nolako harrera egingo zion* is the argument of the nominalized verb asmatu, and another finite verb *zion 'she/he did to her/him'* is its argument instead of being a barrier.

In the following case on the other hand, the subclause marker '-ela' tells that the arguments of asmatu cannot be outside the subclause and the following auxiliary and main verb are barriers to the span of potential arguments.

(4) Ez duzu-la asmatu esan-go dizute, baina badaezpada galdetu egiten du aurretik.
Not do-SUBJ.2.SG.OBJ.3.SG.-SUBCLAUSE guess tell-FUT do-SUBJ.3.PL.OBJ.3.SG.IOBJ.2.SG., but just in case he/she ask him/her beforehand
They will tell you that you have not guessed it, but just in case he/she ask him/her beforehand

Rules can therefore be improved by taking into consideration possible differences in restricting contexts when nominalizations are being used or auxiliaries are missing. Without a dependency annotation of the text, barriers need to be carefully chosen and take into account possible subclauses and clausal arguments of (nominalized) verbs, and they need to distinguish between the two contexts.

4 Conclusion and future work

The experiment has shown that high-level grammar resources encoding deep linguistic analysis such as verb valency information can be reused even for unrelated languages (such as Euskara and North Sámi) and do not need to be built from scratch. Even though language specific adaptions with regard to syntax and morphology need to be made, semantic role specifications can mostly be transferred without changes. Verb valency information is necessary for both linguistically based disambiguation and machine translation tasks.

North Sámi constraint grammar disambiguation rules that make reference to valency information and semantic sets served as a model for developing Euskara disambiguation rules. Grammar rules based on valency frames provide an efficient way to reduce syntactic ambiguity as they manage to select the correct syntactic function in cases where the syntactic context itself remains ambiguous, but the argument specifications of the verb resolve this ambiguity. In machine translation on the other hand, syntactic transfer involving valency-dependent case realizations of the verb's argument can be accomplished by means of linguistic rules that have access to valency information. Additionally, we have seen that polysemy is frequently related to a distinction in valency, which is why valency information has a key function in picking out the correct argument realization and selecting the correct lexical variant in the target language.

Developing parallel resources for two distinctive and unrelated resources does not only benefit NLP, we gain insights in contrastive grammar of understudied languages in general, and the work can serve as a model for the devolopment of linguistic resources for other languages.

Future plans involve extending both the resources and linguistic rules for disambiguation and machine translation. We want to annotate more verbs with valency specifications, which existing general rules apply to, and evaluate the results and improvements. Automatic dependency annotation and semantic role labelling are currently under development and will not only serve the development of grammar rules including valencies, but also benefit from it. Inducing valencies automatically and thereby extending valency resources is another future task.

5 Acknowledgements

The research of this project has been supported by the Department of Education, Universities and Research of the Basque Government (IT344-10) and University (UPV/EHU) (GIU09/19), Giellatekno (Sámi language technology) at the University in Tromsø and the NILS mobility project (Universidad Complutense de Madrid). We would also like to thank Francis Tyers for his helpful critical remarks and corrections.

References

Agirre, E., A. Atutxa, G. Labaka, M. Lersundi, A. Mayor and K. Sarasola (2009), Use of rich linguistic information to translate prepositions and grammar cases to basque, *in* L.Màrquez and H.Somers, eds, 'BEST PAPER AWARD of the XIII Conference of the European Association for Machine Translation EAMT 2009', Barcelona, pp. 58–65.

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., I. Etxeberria, M. Hulden and M. Maritxalar (2010), 'Porting basque morphological grammars to foma, an open-source tool', *Finite-State Methods and Natural Language Processing Lecture Notes in Computer Science* **6062**, 105–113.

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

Beesley, Kenneth R. and Lauri Karttunen (2003), *Finite State Morphology*, CSLI publications in Computational Linguistics, USA.

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

Bick, Eckhard and Pilar Valverde (2009), Automatic semantic role annotation for spanish, *in* 'Proceedings of NODALIDA 2009', Vol. 4 of *NEALT Proceedings Series*, Tartu University Library, Tartu, pp. 215–218.

Forcada, Mikel L., Francis M. Tyers and Gema Ramírez-Sánchez (2009), The free/open-source machine translation platform Apertium: Five years on, *in* F. T.J.A. Pérez-Ortiz, F. Sánchez-Martínez, ed., 'Proceedings of theFirst International Workshop on Free/Open-Source Rule-Based Machine Translation FreeRBMT'09', pp. 3–10.

Hajič, Jan, Jarmila Panevová, Zdeňka Urešová, Alevtina Bémová and Petr Pajas (2003), Pdt-vallex: Creating a large-coverage valency lexicon for treebank annotation, *in* 'In: Proceedings of The Second Workshop on Treebanks and Linguistic Theories', Vaxjo University Press, pp. 57–68.

Karlsson, Fred (2006), *Constraint Grammar - A Language-Independent System for Parsing Unrestricted Text*, Mouton de Gruyter, Berlin.

Kipper, Karin, Anna Korhonen, Neville Ryant and Martha Palmer (2006), Extending verbnet with novel classes, *in* 'Proceedings of the 5th International Conference on Language Resources and Evaluation, Genoa, Italy'.

Lindén, K., M. Silfverberg and T. Pirinen (2009), Hfst tools for morphology—an efficient open-source pacakge for construction of morphological analyzers, *in* 'Proceedings of Workshop on Systems and Frameworks for Computational Morphology', Zürich, Switzerland.

Mayor, Aingeru, Iñaki Alegria, Arantza Díaz de Ilarraza, Gorka Labaka, Mikel Lersundi and Kepa Sarasola (2011), 'Matxin, an open-source rule-based machine translation system for basque', *Machine Translation Journal (to appear)* .

Palmer, Martha, Paul Kingsbury and Daniel Gildea (2005), 'The proposition bank: An annotated corpus of semantic roles', *Computational Linguistics* **31**.

Trosterud, T. and L. Wiechetek (2007), 'Disambiguering av homonymi i Nord- og Lulesamisk', *Suomalais-Ugrilaisen Seuran Toimituksia = Mémoires de la Société Finno-Ougrienne. Sámit, sánit, sátnehámit. Riepmočála Pekka Sammallahtii miessemánu 21. beaivve 2007* **253**, 375–395.

Author Index