Cleaning up the Basque grammar: a work in progress

Inari Listenmaa
University of Gothenburg
inari@chalmers.se

Jose Maria Arriola
University of the Basque Country
josemaria.arriola@ehu.eus

Itziar Aduriz
University of Barcelona
itziar.aduriz@ub.edu

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

1 Introduction

The first version of the Basque Constraint Grammar (BCG) was developed in 1995–1997 by two linguists (Aduriz et al., 1997) based on the Constraint Grammar theory of Karlsson (1990; Karlsson et al. (1995). Since then, it has undergone many changes, by many grammarians. During the two decades of development, the Basque morphological analyser has also been updated several times, and not always synchronised with the CG. As a result, the Basque grammar needs serious attention.

In the present paper, we describe the ongoing process of cleaning up the Basque grammar. We use a variety of tools and methods, ranging from simple string replacements to SAT-based symbolic evaluation, introduced in Listenmaa and Claessen (2016), and grammar tuning by Bick (2013). We present our experiences in combining all these tools, along with a few modest additions to the simpler end of the scale.

2 Previous work

Bick (2013) presents a method for automatically tuning a grammar, and reports an error reduction between 7–15 % when tested on the Danish tagging grammar. Listenmaa and Claessen (2016) present a method for detecting contradictions in a grammar, using SAT-based symbolic evaluation. They report detecting rule conflicts in a few small grammars, but provide no further evaluation on the grammars after fixing the rule conflicts. In our experiments, we use both of these tools for different purposes, complementing each other.

3 Pipeline

As a first step, we run a series of simple, mostly off-the-shelf tools. The next step is to group the rules and order them by their contextual tests. These sets are checked both by the SAT-based tool, and grammarians. After these steps, we give the grammar as an input for ML-tuning.

3.1 Simple tools

String operations Fix typos: O for 0, and various mismatched `"<>"` in word forms: e.g. `"<zuen">`, `<argi>"`. Transform word forms into case-insensitive, remove duplicates. There were many occurrences of identical rules, of the form `REMOVE ("<x>")` and `REMOVE ("<X>")`. We changed those rules into the form `REMOVE ("<x>"i)`, and removed duplicate rules after that.

Tagset operations The VISL CG-3 compiler offers useful features, such as `--show-unused-sets` and `--show-tags`. With the former, we could eliminate 255 unused tagsets, and with the latter, we detected 15 obsolete or misspelled tags in the remaining used tagsets, by comparing against an up-to-date lexical database (Aldezabal et al., 2001).

Human readability For improving the readability of the grammar, we wrote a tool that finds repetitive set definitions, and suggests ways to compact them. An example is shown below:

Original

```
("ageri" ADJ ABS MG)
("bizi" ADJ ABS MG) ...
("haizu" ADJ ABS MG) ;
```

Compact

```
("ageri"|"bizi"|"haizu") +
(ADJ ABS MG) ;
```

In addition, the grammar contains many rules that specify an inline set, when there is already the same or a very similar set definition. For instance, the rule `REMOVE (ADL) IF (0 ADT) (1 ("<.>") OR ("<;>") OR ("<,>") OR ("<:>") OR ("<?>") OR ("<!>"))` lists different punctuation marks as word forms, instead

```
SELECT ADOIN IF (1 ARAZI) ;                          # line 7412
REMOVE ADOIN IF (0 IZE) (1C ADJ) ;                   # line 6423
REMOVE ADOIN IF (0 IZE) (1 DET | ADJ | IZE) ;        # line 6433
REMOVE ADOIN IF (0 EZEZAG + ADJ + GEN) (-2C IZE) ;   # line 6319
REMOVE ADOIN IF (0 IZE) (-1C IZE) (1C ADJ) ;         # line 6422
```

Figure 1: Rules grouped by target, and ordered by their contextual tests.

of using the list PUNTUAZIOA, which contains all these tokens.

The standard tools did not provide this type of suggestions, so we wrote these tools ourselves. Neither of these transformations is applied automatically, they are just suggestions for the grammar writers.

3.2 Group by target, sort by conditions

After the simple checks and transformations, we group the rules by their targets, and sort them by the complexity of their contextual texts. For instance, the 5 rules that target ADOIN will be in the order shown in Figure 1: from fewest to most contextual tests, and in the case of same number of tests, preferring those with fewer tagsets.

3.3 Check for conflicts and redundancies

When the rules are grouped and sorted as described, we run SAT-based symbolic evaluation (Listenmaa and Claessen, 2016) on each group. If it says that some rule with a more complex condition is superfluous because of another rule earlier in the list[1], then that is a hint for the grammar writer: why are there two similar rules in the grammar, if the simpler one would do? At the moment of writing, we are still looking for better ways to adapt the conflict check to the Basque grammar; due to the large number of tags, we cannot use the system straight out of the box. We describe the adaptations we have done so far in Section 4.2, as well as some preliminary results.

3.4 Manual cleanup

Even if the program wouldn't find any conflicts, we give the rules to a grammarian in any case. The grammarian works with this list, having the original grammar on the side to see the comments, or other original context of any given rule. On the one hand, seeing all the rules grouped helps with the situation where different grammarians have written rules independent of each other. On the other hand, working with the ordered grammar makes it difficult to compare the precision, recall and F-score to the original grammar in the intermediate stages of the cleanup.

In any case, the sorting and grouping is not meant to be the final order, it is only to help a human grammarian to make decisions regarding all the rules that target the same tagsets. An easy alternative would be to work on the original grammar instead, only keeping the sorted list as a help and generating new ones as the cleanup proceeds. However, we found it easier to work directly on the sorted list. To solve the problem of intermediate evaluation, we decided to compare the results by ML-tuning both the original and the ordered grammar.

3.5 ML-tuning

Once the grammar has gone through the previous steps, we give it to the ML-tuning tool (Bick, 2013), with the purpose of finding an optimal order. Then we can run the newly ordered grammar through the conflict check, to detect if the ML-tuning has introduced new conflicts or superfluous rules.

Our initial hypothesis is that the human-cleaned version will benefit more from tuning than the original grammar. Some bad rules may have only a minor problem, such as a single tag name having changed meaning, and they would be better fixed by updating the obsolete tag, instead of the whole rule being demoted or killed. To test our assumptions, we tune both the original grammar and the human-cleaned versions, continuously comparing the new versions to the original.

3.6 Final order

After checking the conflicts and redundancies of the grammar, we will proceed to reorder the grammar by defining the sections of the grammar corresponding to each level of granularity of the Basque tag set.

[1] For example, the latter rule of the following is superfluous: `REMOVE Verb IF (-1 Det)` and `REMOVE Verb IF (-1 Det) (1 Verb)`

4 Evaluation

We evaluate the grammars with a manually disambiguated corpus of 65,153 tokens/53,429 words, compiled from different sources (Aduriz et al., 2006). We report the original score, and the result from ML-tuning the original grammar, as well as the result of preliminary cleanup. The scores are given using two metrics, differing on the granularity of the tagset.

4.1 Evaluation criteria

The Basque tag set is structured in four levels of granularity. As explained in Ezeiza et al. (1998), the first level contains only the main POS, 20 distinct tags, and the fourth level contains several hundreds of tags with fine-grained distinctions, including semantic tags such as animacy. Table 1 shows a simplified example of the levels for nouns. On the 4th level, the initial ambiguity is very high: the test corpus has, on average, 3.96 readings per cohort. On the 2nd level, when readings that differ only in higher-level tags are collapsed into one, the initial ambiguity is 2.41 readings per cohort. We follow the scheme for evaluation: assume that we are left with two readings, "Common noun, singular" and "Common noun, plural", and one of them is correct. Evaluation on levels 3 and 4 reports 100 % recall and 50 % precision. Evaluation on levels 1 and 2 ignores the tags from the higher levels, and regards any common noun or noun as correct, hence 100 % for both measures.

It should be noted that the linguistic revision has been targeted towards improving the 2nd level.

4.2 Analysis of the results

The results of the preliminary evaluation are in Table 2. The drop in performance after the preliminary cleanup is most certainly due to ordering—we found it easier to work on the grammar directly after grouping and sorting the rules, as shown in Figure 1. ML-tuning the cleaned grammar brings the precision up, indicating that more rules get to fire in the tuned order. The difference is most dramatic in the sorted and grouped grammar on the 4th level: the original precision drops from 62 % to 56 %, and goes up to 68 % with the ML-tuning.

As explained in Section 3.4, the fairest test at this stage is to compare the ML-tuned results of the original and the cleaned grammar. We see the cleaned and tuned grammar slightly outperforming the tuned original; the difference is not large, but we see it as a promising start.

Conflict check Our main problem is the size of the tag set: all possible combinations of tags on level 4 amount to millions of readings, and that would make the SAT-problems too big. We cannot just ignore all tags beyond level 2 or 3, because many of the rules rely on them as contexts.

As a first approximation, we have created a reduced set of 21000 readings, which allows the program to check up to 200 rules at a time before running out of memory. We are still developing better solutions, and have not run the whole grammar with this setup. Among the first rules we tested, it has found a few redundancies, such as the following:

```
SELECT ADB IF
  (0 POSTPOSIZIOAK-9)
  (-1 IZE-DET-IOR-ADJ-ELI-SIG + INE) ;
SELECT ADB IF
  (0 ("<barrena>")) (-1 INE) ;
```

The problem is that the set `POSTPOSIZIOAK-9` contains the word form "barrena", and the other set contains the tag `INE`; in other words, the latter rule is fully contained in the first rule and hence redundant.

Our second strategy is to reduce the rules themselves: from a rule such as `SELECT Verb + Sg IF (1 Noun + Sg)`, we just remove all tags higher than level 2, resulting in `SELECT Verb IF (1 Noun)`. We also keep all lexical tags intact, but unlike in Listenmaa and Claessen (2016), we allow them to attach to any morphological tags; this may lead to further false negatives, but reduces the size of the SAT-problem. This setup analyses the whole grammar, in the given order, in approximately 1 hour. With the reduced rules, the program would not find the redundancy described earlier, because the problem lies in the 3rd-level tag `INE`. But this approximation found successfully 11 duplicates or near-duplicates in the whole grammar, such as the following:

```
SELECT IZE IF   # line 817
  (0 POSTPOSIZIOAK-10IZE LINK 0 IZE_ABS_MG)
  (-1 IZE-DET-IOR-ADJ-ELI-SIG + GEN) ;
SELECT IZE IF   # line 829
  (0 POSTPOSIZIOAK-10IZE + IZE_ABS_MG)
  (-1 IZE-DET-IOR-ADJ-ELI-SIG + GEN) ;
```

Both of the contextual tests contain 3rd-level tags (`ABS, MG, GEN`), but removing them keeps the

Level 1	Level 2	Level 3	Level 4
Noun	Common noun Proper noun	Common noun, plural absolutive Common noun, singular ergative Proper noun, plural absolutive Proper noun, singular ergative	Common noun, plural absolutive, animate Common noun, plural absolutive, inanimate ... Proper noun, singular ergative, animate Proper noun, singular ergative, inanimate

Table 1: Levels of granularity

	All tags (Level 4)			**48 main categories** (Level 2)		
	Rec.	*Prec.*	*F-score*	*Rec.*	*Prec.*	*F-score*
Original grammar	**95.61**	62.99	75.94	**97.48**	84.37	90.45
ML-tuned original	93.87	68.06	78.91	96.66	86.82	91.48
Preliminary cleanup	94.81	56.56	70.85	96.82	84.13	90.03
ML-tuned prel.cl.	93.41	**68.61**	**79.11**	96.41	**87.19**	**91.57**

Table 2: Preliminary evaluation on words, excluding punctuation, for levels 4 and 2 of granularity.

sets identical, hence it is not a problem for the conflict check.

Finally, all setups have found some internal conflicts. In order to get a more reliable account, we would need more accurate tagset, beyond the 21000. To be fair, many internal conflicts can be detected by simpler means: using STRICT-TAGS would reveal illegal tags, which are the reason for a large number of internal conflicts. But some cases are due to a mistake in logic, rather than a typo; examples such as the following were easily found by the tool.

```
REMOVE ADI IF (NOT 0 ADI) (1 BAT) ;
SELECT ADI IF (0C ADI LINK 0 IZE) ;
```

The first rule is clearly an error; it is impossible to remove an ADI from a reading that does not have one. The conflict likely stems from a confusion between `NOT X` and `(*) - X`. The second rule is not obvious to the eye; the interplay of `0C` and `LINK 0` requires `ADI` and `IZE` in the same reading, which is not possible[2].

ML-tuning So far, the most important use of the ML-tuning has been to overcome the differences in ordering. Given the preliminary nature of the work, we have not tried multiple variations. We used a development corpus of 61,524 tokens and a test corpus of 65,153 tokens; the same which we used to obtain the scores in Table 2. We stopped the tuning after 5 iterations, and used an error threshold of 25 % to consider a rule as "good" or "bad".

[2]ADI is a verb, IZE is a noun.

In the future, as the grammar cleanup advances, we are interested in trying out different settings. Already in our current stage, ML-tuning has clearly improved the precision, for both original and preliminarily cleaned grammars, and for both levels of granularity; it is likely that experimenting with different parameters, we would find a combination that would also improve the recall, like Bick (2013) and Bick et al. (2015) report. However, while ML-tuning improves the grammar's performance, it makes it less readable for human eyes, and continuing the development is harder. Thus we might settle to two versions of the grammar: one for maintenance, and other for running.

5 Future work

After checking the soundness of the grammar by means of some simple tools, we are aware that in the near future we will need more complex utilities for helping the grammar writing. The following items are on our wish list:

Flexible rule ordering We would like the option to view the grammar in a variety of orders, possibly implemented as a feature in the CG IDE. The base order would be one that is easily maintainable and linguistically motivated, and any other orders can be generated from the base order.

Deeper connections between the rules So far we have used the SAT-based conflict check to run the grammar in order, but we would like to develop this further: take any given rule, and give a list of

all the rules, anywhere in the grammar, that potentially feed to it or block it. The biggest problem in developing such a method is the size of the tagset; this leads us to the next item on our wishlist.

Tagset minimisation This feature may be specific to the Basque grammar; for a language with a smaller tagset, there is no reason to restrict the number of tags used in the rules. We propose this idea, because we think it would help to make the SAT-encoding of the Basque grammar more manageable.

The grammar is written to optimize the recall and precision on level 2 tags. It is possible that some of the level 4 or 3 tags used in the rules could be removed without it affecting the functionality of the grammar. Using a development corpus, we could find the minimal set of tags that discriminate between correct and incorrect readings. The following example illustrates the idea:

```
"<lurtarraren>"
  "lurtar" ADJ ARR IZAUR+ GEN
           NUMS MUGM ZERO <Correct!>
  "lurtar" IZE ARR GEN NUMS MUGM ZERO
  "lurtar" ADJ ARR IZAUR+ ABS MG
```

For the given cohort, tags that are only in the correct are `GEN` and only in incorrect are `IZE, ABS, MG`. In other words, we learn that a rule that would target e.g. `ZERO` or `IZAUR+` would not remove all ambiguity. We can compute these tags for all cohorts/ambiguity classes, and see if some tags don't contribute to the disambiguation as much as the others. In such a case, we could simplify the rules in the grammar.

6 Conclusions

We have set out to improve the readability and performance of the Basque CG. The work is in progress, and the improvements on the performance are so far quite minor, but we feel this as a promising start, and a useful case study, for trying out the resources developed within the CG community.

Acknowledgments

This work has been supported by the project UPV/EHU taldea. UPV/EHU (GIU16/16)

References

Itziar Aduriz, José María Arriola, Xabier Artola, Arantza Diaz de Ilarraza, Koldo Gojenola, and Montse Maritxalar. 1997. Morphosyntactic disambiguation for basque based on the constraint grammar formalism. In *Proceedings of Recent Advances in NLP (RANLP97)*.

Itziar Aduriz, Maria Jess Aranzabe, Jose Maria Arriola, Aitziber Atutxa, Arantza Diaz de Ilarraza, Nerea Ezeiza, Koldo Gojenola, Maite Oronoz, Aitor Soroa, and Ruben Urizar. 2006. Methodology and steps towards the construction of EPEC, a corpus of written Basque tagged at morphological and syntactic levels for the automatic processing. In *Corpus Linguistics Around the World*, volume 56 of *Language and Computers*, pages 1–15. Rodopi, Netherlands.

Izaskun Aldezabal, Olatz Ansa, Bertol Arrieta, Xabier Artola, Aitzol Ezeiza, Gregorio Hernndez, and Mikel Lersundi. 2001. Edbl: a general lexical basis for the automatic processing of basque. In *IRCS Workshop on linguistic databases. Philadelphia (USA)*.

Eckhard Bick, Kristin Hagen, and Anders Nklestad, 2015. *Optimizing the Oslo-Bergen Tagger*, pages 11–19. Linkping University Electronic Press.

Eckhard Bick. 2013. ML-Tuned Constraint Grammars. In *Proceedings of the 27th Pacific Asia Conference on Language, Information and Computation (PACLIC 2013)*, pages 440–449.

Nerea Ezeiza, Itziar Aduriz, Iñaki Alegria, Jose Mari Arriola, and Ruben Urizar. 1998. Combining stochastic and rule-based methods for disambiguation in agglutinative languages. In *COLING-ACL'98. Pgs. 380 - 384. Vol 1. Montreal (Canada). August 10-14, 1998*.

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

Fred Karlsson. 1990. Constraint grammar as a framework for parsing running text. In *Proceedings of 13th International Conference on Computational Linguistics (COLING 1990)*, volume 3, pages 168–173, Stroudsburg, PA, USA. Association for Computational Linguistics.

Inari Listenmaa and Koen Claessen. 2016. Analysing Constraint Grammars with a SAT-solver. In *Proceedings of the 10th edition of the Language Resources and Evaluation Conference (LREC 2016)*.