ClariaH
ACAD project
Deliverable D13 and D15
Evaluation of Clariah components
and
suggestions for improvements

Erwin R. Komen
TSG, Radboud University Nijmegen

The ClariaH project "Automatic Coherence Annotation for Dutch" (ACAD) has made use of a number of ClariaH components. Some of these are text corpora (the SoNaR part of Lassy-Groot, VU-DNC, CGN) and some of these are software applications (Alpino, CorpusStudio web application). This combined D13-D15 deliverable reviews the components that have been used, it shows where improvements have already been made, and it contains a number of suggestions for improvements that have not yet been implemented.

1 Corpora

The text corpora that are used from withing the ACAD web application 'Cesar' all need to be in the FoLiA *xml* format, and all of them need to have either an *imdi* or else a Sonar *cmdi* metadata definition.

It is not only the particular format that is important. The Cesar program assumes that the syntactic annotation is 'surfaced', which means that the hierarchical order of the constituents agrees with the linear surface order.

1.1 The SoNaR part of Lassy-Groot

The Lassy-Groot corpus consists of a large number of texts that have been syntactically parsed and that are available in the Alpino_ds *xml* format. One section of these texts is the syntactic parse of the SoNaR treebank. This is the section that has been used by the ACAD web application 'Cesar'.

The SoNaR corpus consists of a number of sections itself (e.g. legal texts, electronic and paper newsletters). Most of the Sonar-sections were already 'surfaced' (see above) for the CorpusStudio web application with the FoliaParse utility. The following sections were added for the ACAD project:

Sonar code	Contents	Size
WR-P-P-G	newspapers	82 Gb
WR-U-E-A	chats	2.1 Gb
WR-U-E-D	sms (short text messages)	98 Mb

Most of the syntactic parses that have in the past been made with Alpino were usable for the ACAD purposes. The parses from the SMS corpus were less usable (they have not been used in real searches). The reason for this is not the Alpino parser itself, but the type of language that is used in text messages. That languages many times misses the normal clues that help construct the grammatical layout of sentences (such as overt subjects and objects).

One thing that gave us some difficulties and that should be mentioed here is the flexibility of the FoLiA format itself. This is not to be taken as criticism; it is something to be aware of.

We found two FoLiA flexibility matters that took us some time to resolve: (i) the fact that <w> tags do not necessarily are direct children of the <s> node (other tags can intervene, grouping several <w> tags together), and (ii) the fact that a <w> tag can have multiple <lemma> nodes.

1.2 The VU-DNC

The VU-DNC is a text corpus of articles taken from the major Dutch national newspapers. It contains articles from each of those newspapers taken from two years: 1951 and 2002. The corpus as we received it had metadata in the appropriate .cmdi *xml* format, and it had POS-tagged FoLiA *xml* files. What it lacked was the syntactical parse. Micha Hulsbosch took it upon him to add that layer to the files, making use of the FoliaParse surfacing program, which, in turn, makes use of the Alpino parser to determine the syntax. The Alpino parser is able to do POS-tagging itself, but the setup of FoliaParse is such, that the POS-tags of the existing FoLiA are used as a basis.

1.3 The Corpus of spoken Dutch (CGN)

This corpus had already been turned into FoLiA and surface-converted for the CorpusStudio web application project (CLARIN-NL). While working with it, however, we found two things: (i) the POS-tags that are used differ from the Sonar set of tags, and (ii) the lemma's were missing.

Circumventing the POS-tags was not too difficult. That just meant that the linguist-user needed to define the syntactic search functions slightly different. The only consequences were in parts of searches that triggered particular adjectives or verbs. Changing the strings that define those constituent types was all that needed to be done.

Patrick Sonsma, the student-assistant working under Wilbert Spooren on the ACAD project, kindly took it on him to search for the lemma's in the CGN, and, with a little help from Nelleke van Oostdijk, he found where they were hiding. He then made them available at the appropriate locations (as children of <w>) in the FoLiA *xml* of the CGN that we had.

1.4 Suggestions for improvements

All of the corpora above come with metadata. What linguists usually want to know is the number of words (not tokens; just lexical words) that are in a text. It would be good if the metadata could be extended with this information, and it would be good if this kind of information were added as a standard to any new text corpus.

2 Software applications

2.1 Alpino

The 'Ponyland' Linux computers have a working configuration of the Alpino parser, and it is this version that has been used to perform all syntactic parsing of the texts as described in section 1. We are very pleased with the syntactic results of the Alpino parser. It is not a very fast parser, and we found that its processing time more than increases with the length of the sentence.

Improvements: it would be great if the Alpino parser would get a speed improvement.

2.2 The CorpusStudio web application

The CorpusStudio web application has been developed as a CLARIN-NL project. It consists of two parts: (i) a user interface 'CrpStudio' and (ii) a back-end 'Crpp'. Both parts are web applications and they need not run on the same machine. They communicate with one another through an API which is well described in the CorpusStudio API documentation.

We did not use the CrpStudio user interface for the ACAD project, since the whole idea of this project is to come up with a more linguist-friendly interface. The existing CrpStudio interface is meant for those researchers that are able to define searches in the Xquery language.

What we did make use of was the 'Crpp' back-end, and we have made several improvements in the course of the ACAD project.

1) Extensions of existing commands:
   a) /txt – added the possibility to get all the 'sentences' of a text
   b) /dbinfo – filtering and sorting of the results

2) Newly added commands:
   a) /txtlist – get a list of available texts of a language/part combination.
      i) Each text is accompanied by a small set of most relevant metadata information, which is extracted from the corresponding CMDI or IMDI *xml* files.
      ii) The metadata information should also contain the **size** of the text in number of actual words (excluding punctuation). We have not yet calculated the correct sizes for all texts, this remains a follow-up point.

3) Reaction time improvements
   a) The speed of locating any text file has increased tremendously by pre-calculation of text directories on the basis of the Language-Part combinations.

4) Space improvements
   a) The back-end is now able to work with gzipped text files. This proved to be very helpful to handle the WR-P-P-G newspaper corpus better.

5) Result extensions
   a) The original Crpp made results available in an *xml* database file. We added two other formats: an SQlite database file and a CSV text file.
   b) The SQlite file is used to facilitate filtering and ordering results in the back-end (as opposed to using the Cesar front-end).
   c) The CSV file is made 'on-the-fly' to facilitate fast and easy downloading of query results. We are assuming that linguists make searches and want to process the results further on their own computer. One way to do this is by downloading the results in CSV.

Improvements:
Word sizes. The size in words should be re-calculated where necessary, and then made available.
Speed and location. The speed of the back-end is manageable, but it would be very helpful if it were tuned up. As for the location of the back-end: that needs improvement. Its location right now is in a SurfSara virtual machine. The problem with that is that there is a frequent and unpredictable downtime. There also is a restriction on the speed.

2.3 FoliaParse

We had not entered the FoliaParse software program as a ClariaH component to be used in the ACAD application, but we have made extensive use of it during our project. It has proved its (unique) value in converting Alpino parses into a surface form that is easier searchable by the Xquery code used in CorpusStudio.

We had several bugfixes and one larger improvement in the FoliaParse program. The larger improvement has to do with the way 'unparsed' words are treated. Alpino does not always find a syntactic position for all words, and when it doesn't it adds the words at the top level. The

improvement we made to FoliaParse positions these words on the correct (surface-taken) locations in the syntax tree.

Improvements:
The internal make-up of FoliaParse could do with a complete makeover. The current code first makes a transformation to the psdx format, does the surfacing, and then puts the <su> units into place. The psdx-transformation might better be circumventend, so that surfacing is done more straightforwardly.

A second point is that the FoliaParse program only provides one <syntax> layer: the one with the surfaced results. It would be better to provide two layers: one with the 'original' Alpino structure (which is not necessarily surface-correct) and one with the surfaced layer. The layers should be differentiated by adding a @class attribute to the <syntax> layers, which is the FoLiA method to deal with this.

A third point is that we had a unicode translation problem somewhere, but we were not able to determine the exact reason for it (issue #74).