Universiteit Utrecht

Provenance in AnnCor & CLARIAH WP3

Jan Odijk

DANS, Den Haag, 2018-09-03

Utrecht Institute of Linguistics OTS

Overview

- Input data
- Upload into GrETEL 4
- Querying in GrETEL 4
- Analysis of Search results
- MOR and GRA tiers
- Manual Editing
- General Questions and Remarks

Input Data

- set of chat files (*.cha); CHILDES
- **no** version number
- and **no way to specify** a version number
  - in the header or in the metadata.
    - DC, OLAC: no *version* attribute, they do have *date{Submitted/Accepted/CopyRighted}*. (no *edition* attribute either)
    - CMDI has *version* but always 'unknown'. E.g. this one and this one

Input Data

- Brian MacWhinney:

  We could conceivably add this, but if we did, we would have to add it to everything. Moreover, we would then have to keep a zillion versions on the web and in VLO etc., because we are continually making corrections and additions of this type to the corpora. Instead of going this way, we are relying on GIT repository archiving to pull back any old versions that people need.

See also: https://talkbank.org/info/versions.html

Upload into GrETEL 4

- Convert to metadata-annotated plain text file: *chamd.py* **version**
  - Converts in-line annotations (2 options) (*cleanCHILDESMD.py*) **version, parameter**
- Alpino-parser (**version**, **parameters**)
- Leads to searchable and **downloadable** treebank for the chat files in XML format
  - Following the Alpino DTD (**version, DTD location**)

Querying in GrETEL 4

- Search results (matches: list (treebank sentence, match information):
  - GrETEL **version**, **query, source treebank, source treebank version**
- New corpus based on a query:
  - GrETEL **version**, **query, source treebank, source treebank version**

Analysis of Search results

- Yields a pivot table, downloadable
  - GrETEL **version**, **query**, **source treebank**, **source treebank version**, **analysis options selected**

MOR and GRA tiers

- Create enriched CHAT corpus with MOR (morphology) and GRA (syntax) tiers
- Generate MOR-tier:

  vlgaan-IMP-SG advlmaar advleven vlliggen-INF

  – Uses programme (**name**, **version**)
  – Uses mapping table (**name**, **location**, **version**)
  – Two styles, indicated by a **parameter**

MOR and GRA tiers

- Generate GRA tier
- E.g.

1:ga 2:maar 3:even 4:liggen

%GRA: 1|0|ROOT 2|4|mod 3|4|mod 4|1|vc

  - Uses programme (**name**, **version**)

Manual Editing (1)

- For each utterance and its syntactic structures edit:
  - [Optional] Utterance Transcription Correction
    - Correction of the transcription
      - *das goed -> da(t) (i)s goed*
      - Original utterance is retained
      - (previous utterance is retained)
    - Re-cleaning of the utterance

Manual Editing (2)

– [Optional] Bracketed Input
  - Add bracketing to the cleaned utterance, e.g.

    Ik wil [@skip xxx] [@pp naar huis]
  - Store bracketed input as Alpino input metadata
  - Parse using brackets if available in the input
    – Alpino-parser: **name**, **version**, **location**, **parameters**, …

Manual Editing (3)

- Edit syntactic structures manually
  - Using *TrED* (**name**, **location, version**)
- Apply *dtcanonicalize* (**name**, **location**, **version, modules + versions it depends on**)
- Apply *ACE* (AnnCor Check Engine) to the syntactic structure—
  - Yields errors and warnings in an output file
  - Some errors and warnings can be overruled by the annotator in the output file

Manual Editing (4)

– Publish the parsed utterance, with the **annotated ACE-output**, with **anonymized annotator IDs**

General Questions and Remarks

- Data → Tools ➔ new data
  - Tools
    - Are in a programming language: **name, version**, **compiler / interpreter name, version**
    - Use other own modules (**name**, **version, programming language**) recursively!
    - Use external modules (**names**, **versions**, **programming language** …) ..recursively!

General Questions and Remarks

- Data → Tools ➔ new data
  - Tools (cont.)
    - Use data files (**name, version**, …)
    - Use information and software from external `unreliable' sources; Linked Data, URLs, PIDs, web applications, web services, … recursively!
- Storing all this will lead to huge amounts of information. Who will ever inspect them?

General Questions and Remarks

- Where do we store provenance information?
  - In external files?
    - Where should they be put?
    - Danger of losing the connection.
    - Which Data Model? PROV-DM? In which format?
  - Inside the generated files?
    - Sometimes there are options, but not always
    - Could OSs not be adapted for this?
  - Both?

General Questions and Remarks

- Spare the user and the developer!
  - The end user should **never** be bothered by provenance information (only benefit)
  - The application or tool developer should **never or hardly ever** be bothered by ensuring generation of provenance information.
  - The programming language environment should do all (most?) of that in the background. Cf. recipy

General Questions and Remarks

- Origin of differences between different versions of data should be easily (i.e. automatically) detectable
- Ideally, provenance data can be used directly to fully automatically regenerate the output based on the input
  - Cf. differences between texts a la Levenshtein distance also yields a program to change the texts.

Thanks for Your Attention!