From CHAT to Coded CoNLL-U: A Reproducible Pipeline for the Syntactic Annotation and Querying of Child Language Data

Achim Stein
Institute of Linguistics
University of Stuttgart (Germany)
achim.stein@ling.uni-stuttgart.de

Abstract

The CHILDES database is a core resource for language acquisition research, yet its CHAT format poses significant challenges for modern computational analysis. To address this, we present a reproducible, open-source pipeline that transforms CHAT transcripts into annotated tabular (CSV) and CoNLL-U formats. Its core script, childes.py, automates the conversion and integrates part-of-speech tagging and dependency parsing. A key innovation is dql.py, a tool that uses a Grew dependency query language to systematically add user-defined linguistic codings to the parsed data. While the script is parametrised for various languages, the pipeline's utility is demonstrated by applying it to the French CHILDES corpus to conduct a large-scale analysis of object clitic production. The resulting structured data reveals clear developmental trajectories, such as the gradual convergence of children's dative clitic usage towards the adult input. The workflow and the resources it generates facilitate reproducible, data-driven research in language acquisition.

Keywords: Language Resources, CHILDES, CHAT, CoNLL-U, Dependency Parsing, Corpus Querying, Language Acquisition, French

1. Introduction

The Child Language Data Exchange System (CHILDES) database is a cornerstone of language acquisition research, providing access to a large collection of transcribed conversational data (MacWhinney, 2000). The data are formatted in the CHAT (Codes for the Human Analysis of Transcripts) convention (MacWhinney, 2019), a rich system designed for human readability and analysis with the dedicated CLAN software suite. While powerful, the CHAT format and the CLAN environment present significant hurdles for researchers wishing to apply modern Natural Language Processing (NLP) tools or conduct large-scale quantitative analyses in statistical environments like R or Python. The process of converting CHAT files to standard tabular or annotated formats is non-trivial, often leading to ad-hoc solutions that compromise reproducibility and data integrity.

To bridge this gap, we present a complete, open-source pipeline that transforms CHILDES data from its raw CHAT format into richly annotated, computationally tractable resources. Our workflow produces two primary output formats: comprehensive tabular (CSV) files in a one-word-per-line format, and syntactically parsed CoNLL-U files. This process preserves crucial metadata while adding multiple layers of linguistic annotation, including part-of-speech tags, lemmas, and dependency syntax.

The pipeline relies on two core Python scripts. The main script, `childes.py`, manages the end-to-end conversion from CHAT to annotated CSV and CoNLL-U by integrating standard tools like the UDPipe parsing API (Straka, 2018, 2023) and, optionally, TreeTagger (Schmid, 1994). A second script, `dql.py`, provides a querying layer, using the Grew query language (Guillaume, 2019; Guibon et al., 2020) to search the parsed dependency graphs for specific linguistic constructions and add user-defined codings to the output files. While integrated into the workflow described here, both scripts are standalone applications for converting CHAT data and querying CoNLL-U corpora, respectively.

This paper details the architecture of our pipeline (Section 2), describes the resulting resources and their availability (Section 3), presents a case study (Section 4) and concludes with a discussion of limitations and future work (Section 5).

2. The Annotation Pipeline

Our pipeline is designed as a modular workflow that automates the conversion and annotation of CHAT files. The architecture proceeds from raw CHAT data to a fully coded tabular dataset ready for statistical analysis.

2.1. Architecture Overview

The pipeline is initiated via a shell script (`childes-pipeline.sh`), which executes the main Python script, `childes.py`. The process is visualised in the flowchart in Figure 1.

1. **Conversion:** `childes.py` reads the input CHAT file, retrieves its metadata, parses its conversational structure, and tokenises each utterance into an internal one-word-per-line format.

Proceedings of the Fifteenth Language Resources and Evaluation Conference (LREC 2026), pages 11516–11523
11–16 May 2026. ©ELRA Language Resources Association (ELRA), 2026

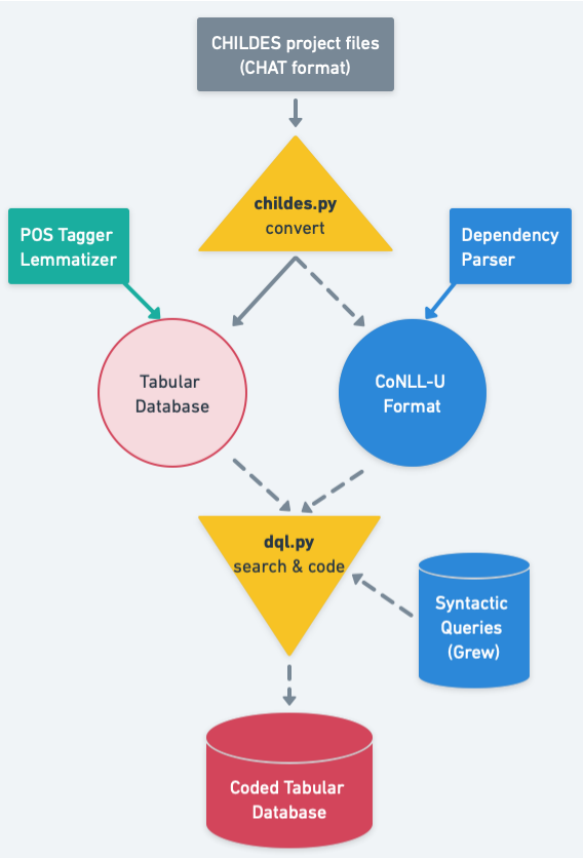

Figure 1: The CHAT processing and annotation pipeline, from raw transcript to queryable CoNLL-U and analysable CSV formats.

2. **Annotation Management:** The script then manages linguistic annotation by interfacing with external tools based on the provided command-line arguments:
   - If a TreeTagger parameter file is provided (`--parameters`), the script sends the tokenized text to a local TreeTagger instance for part-of-speech (POS) tagging and lemmatization.
   - If a UDPipe model is specified (`--api_model`), the script prepares the data (raw tokens or TreeTagger's output) in CoNLL-U format and sends it to the UDPipe web API for dependency parsing.
3. **Multi-format output generation:** Using the data gathered from the previous steps, `childes.py` generates a set of output files in a single run:
   - The full tabular database file (`.parsed.csv`) with all available data, including CoNLL-U columns, as well as a smaller `.light.csv` file where rows can be filtered by POS tags.
   - A standard CoNLL-U file (`.conllu`) containing the full dependency parse information for further analysis.
   - Optionally, a collection of interlinked HTML files that provide a browsable interface for visualizing the dependency trees (not shown in the workflow, see Figure 4 for an example).
4. **Optional linguistic coding:** Finally, the separate `dql.py` script can be run on the generated `.conllu` file. This script uses graph-based Grew queries to add specific linguistic codes, producing an enriched `.coded.conllu` file. These codes can then be merged back into the main CSV table.

These steps are described in more detail in the following sections, and exemplified in the use case of Section 4.

2.2. Core Processing with `childes.py`

The central component, `childes.py`, is a robust CHAT parser and data integration tool. In its most basic mode, the script can be used simply to convert CHAT files to a tabular, one-word-per-line format without any further linguistic annotation. However, we focus here on a more comprehensive workflow that showcases the script's ability to integrate external NLP tools for tagging and parsing. This process is controlled by a variety of options, a full discussion of which is beyond the scope of this paper (for complete documentation, see the project's GitHub repository [link upon acceptance]).

CHAT Parsing. The script processes CHAT files in a stream-based manner, making it efficient for large, concatenated corpora. It correctly parses essential CHAT components, including headers (`@ID`, `@Participants`) to extract speaker metadata (age, role, etc.) and the main tiers (`*`) for utterances, and timestamps. Crucially, it preserves links to the original source by generating a unique ID (`utt_id`) for each token based on the file's PID and utterance/word numbers.

Linguistic Annotation Layers. Once the data is in a tabular format, `childes.py` adds several layers of annotation. The pipeline is designed for flexibility, allowing users to specify different models for tagging and parsing via command-line arguments (e.g., `--parameters`, `--api_model`).

- **Tokenisation and Cleaning:** The script's internal tokeniser, similar in function to the one distributed with TreeTagger, is designed for adaptability. It contains conditional, language-specific blocks that are activated based on the

language specified in the CHAT file's header. For French, it handles context-dependent phenomena such as the apostrophe (separator with French *l'* as clitic or article, but part of the word with *aujourd'hui*). The cleaning process also handles CHAT's special symbols such as those for unintelligible speech (`xxx`), phonological fragments (`&=...`), retracings (`[//]`, and others: while a cleaned version of the utterance is fed to the NLP tools, the raw version including CHAT coding is preserved in the tabular output, ensuring that no information is silently dropped.

- **Dependency Parsing:** To obtain syntactic structures, the script formats the tokenised text as a CoNLL-U input file and sends it to the UDPipe API. To handle large files and prevent API timeouts, the data is automatically sent in configurable chunks (`--chunk_parse`). 10,000 utterances have proven to be a safe chunk size and are processed in under 30 seconds by the API. The script also includes a debugging routine that, upon an API failure, re-submits the failing chunk in smaller segments to precisely identify any malformed utterance that may be causing the parser to fail.

  Furthermore, the script supports an optional rewriting stage (`--rewrite`), which applies a Grew Graph Rewriting System (GRS) to the parser's output to correct systematic errors or adapt the annotation to specific project guidelines before the final CoNLL-U output is generated. The distribution (version 3.0) includes sample rewriting rules for French, which can be adapted or extended by users for their specific needs. They address some systematic issues with incomplete utterances and non-canonical word orders, which are common in child language data.

- **Part-of-Speech and Lemma Annotation:** Optionally, the pipeline integrates TreeTagger for additional POS tagging and lemmatisation. Users can specify any TreeTagger parameter file appropriate for the target language and task. If this additional analysis is included, the coding queries described in Section 2.3 can refer to either the parser's or the TreeTagger's POS and lemma annotations.

While the current implementation uses TreeTagger and UDPipe, the modular structure of `childes.py` is designed to facilitate the straightforward integration of alternative NLP tools for these annotation stages.

2.3. Advanced Querying with `dql.py`

While UDPipe provides a general-purpose syntactic analysis, targeted research questions often require identifying specific, theory-relevant constructions. Our `dql.py` script serves this purpose by adding a user-defined coding layer on top of the parsed CoNLL-U data.

The script uses the Grew query engine and the 'grewpy' Python module to search dependency graphs for patterns specified by the user. A key feature of `dql.py` is its ability to process multiple Grew queries sequentially from a single file containing the requests and coding instructions. This batch-like functionality was inspired by the coding function of *CorpusSearch*, a tool widely used for querying corpora parsed in the Penn Treebank style (Randall et al., 2004).

In the request file, each Grew pattern is preceded by a coding comment line (`% coding`) that specifies a coding instruction, defining an attribute and a value (see Section 4 for an example). When a pattern matches a sentence in the CoNLL-U file, `dql.py` adds the corresponding attribute-value pair directly as a new line to that sentence's CoNLL-U metadata block. This enriched CoNLL-U file, now containing explicit annotations for multiple, user-defined linguistic features, serves as the final input for the merge step that creates the fully coded CSV table, where each coding attribute becomes a new column. This method allows researchers to systematically and reproducibly annotate a wide range of grammatical phenomena in a single pass.

Figure 2 shows one of the four requests (or queries) that we will use in Section 4. It identifies a personal pronoun (third-person clitic) in an indirect object dependency relation 'iobj'. Apart from the coding comment line (which is printed on two lines in our figure), patterns follow the regular Grew syntax and can optionally combine the positive 'pattern' statement with a negative 'without' statement (see `https://grew.fr` for a detailed Grew documentation).

```
% coding attribute=clitic_dat
  ...value=iobj3 node=PRO addlemma=V
pattern {
   V -[iobj]-> PRO;
   V [upos=/AUX|VERB/];
   PRO [form=/^[lL].*/, PronType="Prs"];
  }

% coding attribute=... [next block]
```

Figure 2: A Grew query to identify third-person dative clitics.

The script 'dql.py' is called in two steps. First, it applies the request file to the CoNLL-U file produced by 'childes.py'. It matches each pattern of

the request file against the dependency graphs in the CoNLL-U file. Upon a match, it adds a coding to the sentence's metadata block, as illustrated in Figure 3. The coding directive specifies an attribute (in our example: 'clitic_dat') and a value ('iobj3') to be assigned upon a match. The coding line also specifies the reference node from the query to which the annotation should be attached. Optionally, `addlemma=NODE` can be used to append the lemma of another node from the query to the value string, providing richer contextual information.

```
# item_id = 42879_u1681
# coding = clitic_acc:obj3(5>7_donner);
...clitic_dat:iobj3(6>7_donner)
1 ben ben INTJ _ _ 3 advmod _ _
2 tu toi PRON _ PronType=Prs 3 nsubj _ _
3 peux pouvoir VERB _ _ 0 root _ _
4 venir venir VERB _ _ 3 xcomp _ _
5 le le PRON _ PronType=Prs 7 obj _ _
6 lui lui PRON _ PronType=Prs 7 iobj _ _
7 donner donner VERB _ _ 4 xcomp _ _
```

Figure 3: A CoNLL-U graph with syntactic codings added by 'dql.py'.

Each matched query block of the request file adds a coding such as 'clitic_dat=iobj3' to the CoNLL-U metadata of the sentence. In Figure 3, two codings were added to the metadata: 'clitic_acc' for a third-person accusative clitic *le* and 'clitic_dat' for a third-person dative clitic *lui* (Figure 2 only shows the pattern matching the dative). The value strings include additional information in parentheses, generated by the `addlemma` directive. In this case, the resulting value appears as 'iobj3(6>7_donner)', where 'iobj3' is the specified value for a 1st/2nd-person indirect object, '6>7' represents the dependency between the nodes, and 'donner' is the lemma of the node specified by `addlemma`. The additional information in parentheses can be useful for debugging or further analysis, or just disposed of if not needed.

In a second step, 'dql.py' is called with the `--merge` flag to integrate these codings from the CoNLL-U metadata into the final CSV table. In the table, the coding attributes become new column headers, and the corresponding values are filled in for each token in the relevant row. If a sentence matches multiple patterns from the request file, multiple codings are added to its metadata and, as columns, to the CSV table. This two-step process allows for a systematic and reproducible annotation of complex linguistic phenomena across large corpora.

3. The Resource: Availability and Format

The primary contribution of this work is not only the pipeline itself but also the enriched language resources it generates. The following components are made publicly available on GitHub under a GPL licence (Stein, 2026):

- **The Pipeline Software:** The source code for `childes.py`, `dql.py`, and the `childes-pipeline.sh` wrapper script are available.
- **Query File:** The Grew request file used in Section 4 demonstrates how to code French object clitics with 'dql.py'.
- **Grew Rewriting Rules:** Sample Grew rewriting rules used for correcting systematic errors in the (French) UDPipe analysis.

For optional additional tagging with the `--parameters` flag, users will have to provide the TreeTagger binary for their operating system and appropriate parameters, both freely available at the TreeTagger website.

The main output format is a tab-separated values file designed for easy import into R, Python (pandas), or spreadsheet software. Each row corresponds to a single token. This format is intentionally redundant; for example, the full utterance text is repeated for each token belonging to it, providing complete context on every line. To manage this redundancy, the script generates two distinct tables. A comprehensive 'full' table (`.parsed.csv`) contains all generated data, including columns for the parsed CoNLL-U representation, allowing for further computational processing. In contrast, a streamlined 'light' table (`.light.csv`) is created for direct analysis. This version contains a reduced set of columns, and its rows can be filtered using command-line options such as `--pos_output`, which restricts the output to tokens with specific part-of-speech tags. The following list details the key columns available in the full table:

- `utt_id`: A unique token identifier (e.g., '42879_u1_w1') that enables linking back to the original CHAT file (PID, utterance number, word number).
- `speaker`: The three-letter speaker code from the CHAT file (e.g., 'CHI' for child, 'BRO' for brother).
- `child_project`: script-generated project identifier based on the child and project name (e.g., 'Antoine_Par' for Antoine from the Paris corpus).

- `age`, `age_days`: age metadata extracted from the CHAT headers, converted to days and added also to the non-child utterances of each recording (to facilitate analysis of child-directed speech).
- `word`: The word or token.
- `lemma`, `pos`: The POS and lemma annotations, optionally added by the tagger or copied from the parser analysis.
- `utterance`: the unmodified utterance, including CHAT-specific coding. Options of the script allow to add a cleaned version without CHAT coding (`--clean_utterance`) and a version with part-of-speech tags (`--tagged_utterance`).
- `conll_*`: Columns containing the full CoNLL-U dependency information (e.g., head ID, dependency relation).
- Optional columns containing coding information, user-defined in the request file of 'dql.py' (e.g., `clitic_acc`, `clitic_dat`).

This rich, tabular format makes the data immediately amenable to a wide range of statistical and computational analyses that are difficult to perform on raw CHAT files. It offers a standardised part-of-speech annotation and lemmatisation, replacing the CHAT annotation that can be absent, idiosyncratic or inconsistent across or within individual projects. The inclusion of both a full and a light version of the table allows users to choose between comprehensive data for in-depth analysis and a streamlined dataset for quick exploration.

In addition to the CSV and CoNLL-U files, 'childes.py' offers an optional HTML output functionality, activated with the `--html_dir` flag. This feature generates a set of browsable HTML files containing visualizations of the parsed dependency trees for each utterance. When this option is used, both the full and the light CSV tables are populated with hyperlink columns (`URLwww` for a remote server and `URLloc` for a local server). The links allow the user to jump from the tabular data to the corresponding utterance in the HTML representation. This allows for efficient inspection of the larger conversational context and of the detailed, color-coded dependency parse, as illustrated in Figure 4, bridging the gap between the condensed table and the full syntactic analysis. In this case of a cleft construction with omitted relative pronoun (translation: "It is the wind [that] makes the noise of a giant.") the context can disambiguate *bise* as 'wind' rather than 'kiss', or help identify the utterance as a partial quote of a poem (by Victor Hugo) rather than an original child production.

ID: 42879_u46828 | Anae_Par | CHI | 4;08.08

```
<c'est la bise fait le bruit d' un géant> [=! récite] .

01.... c' ce PRON  nsubj→4
02.... est être AUX  cop→4
03.... la le DET  det→4
04 bise bise NOUN  root→0
05.... fait faire VERB  acl→4
06.......... le le DET  det→7
07........ bruit bruit NOUN  obj→5
08.............. d' de ADP  case→10
09.............. un un DET  det→10
10.......... géant géant NOUN  nmod→7
11.... . . PUNCT  punct→4
```

Figure 4: HTML dependency parse visualisation, including some utterance metadata, the raw utterance, and the color-coded dependency tree.

4. Use Case: The Production of Object Clitics in French CHILDES

We present a brief case study on the acquisition of object clitics in French, a topic of significant interest in developmental linguistics. Our analysis is motivated by the work of Pirvulescu and Strik (2014), who, among others (e.g., Delage et al. (2016) for French accusatives and Anna Cardinaletti et al. (2021) for Italian) investigated children's *comprehension* of object clitic features. Their experimental study found that French-speaking children, particularly 3-year-olds, struggle to use the gender and number features of object clitics to identify the correct referent, performing significantly worse with clitics than with strong pronouns. They concluded by highlighting the need for research that correlates these comprehension difficulties with patterns in spontaneous *production*. Our pipeline provides the means to generate and analyse the large-scale production data required to address such questions.

As explained in Section 2, `childes.py` transforms the raw CHAT files into a queryable CoNLL-U format. This allows us to use `dql.py` to execute Grew queries that identify different types of object clitics based on their syntactic function and features. For instance, we can distinguish third-person accusative clitics (*le, la, les*) from dative clitics (*lui, leur*).

For our use case, we applied the pipeline to eleven projects from the French CHILDES and PhonBank corpus (CHAT files downloaded from TalkBank; `https://talkbank.org`). Our Grew request file contained four queries to distinguish clitics by case (accusative 'obj' vs. dative 'iobj') and person (third-person vs. first/second-person), since French 1st/2nd person forms are ambiguous for case. The resulting coded tables were imported into R for analysis and visualisation (the script 'childes-lrec2026.Rmd' is provided). To ensure robust developmental comparisons, utterances were aggregated into dynamic age bins, each represent-

ing a comparable amount of linguistic production from children (Figure 5). This data-driven binning approach avoids the biases of fixed chronological intervals.

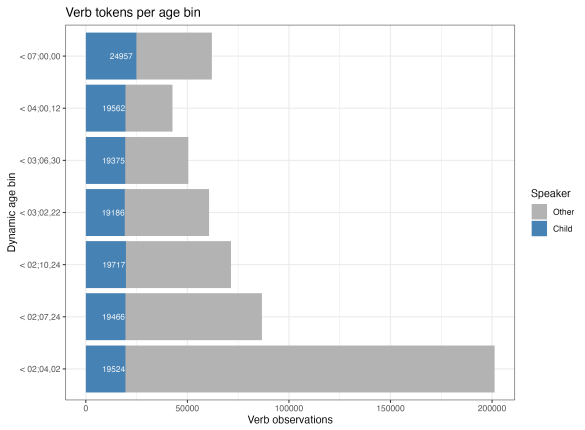

Figure 5: Distribution of verb tokens across the seven dynamic age bins for children (Child) and caretakers (Other). Each bin contains approximately 20,000 child verb tokens.

Figure 6 shows the absolute frequencies of accusative and dative clitics produced by children. A clear asymmetry is evident: accusative clitics are much more frequent than dative clitics across all age bins. Notably, within the infrequent dative constructions, children predominantly use first-person and second-person forms (labeled as 'clitic_12'), while third-person dative clitics ('clitic_3') remain rare throughout early development.

Figure 6: Absolute counts of accusative and dative clitics produced by children, distinguished by person (1st/2nd vs. 3rd) across age bins.

To compare the developmental trajectories directly, we calculated the proportion of dative clitics relative to all object clitics for each age bin and speaker role (children vs caretakers; Figure 7).

Our analysis of the corpus reveals a clear developmental trend. Children's proportional use of

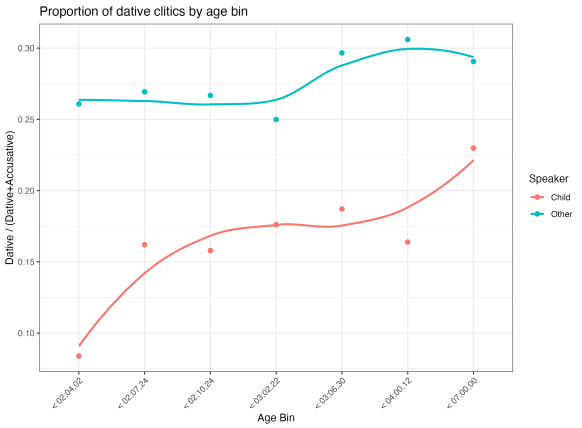

Figure 7: Proportion of dative clitics (relative to all accusative and dative clitics) in child speech ('Child') and child-directed speech ('Other') across dynamic age bins.

dative clitics starts significantly lower than that of adults in their child-directed speech (approx. 9% vs. 26% in the earliest age bin). As shown in the figure, the child's production of datives then steadily increases with age, more than doubling to over 20% in the final bin and thereby converging towards the more stable proportion observed in the input. Tracking such fine-grained syntactic developments on a large scale would be prohibitively difficult without our automated pipeline.

This case study demonstrates how the workflow transforms a vast collection of text-based transcripts into a structured, queryable resource, enabling researchers to efficiently investigate complex questions in language acquisition and provide the large-scale, production-based evidence called for in the literature. This brief analysis, therefore, is not intended as a definitive answer to the research questions raised by Pirvulescu and Strik (2014) and other work. Its purpose is primarily methodological: to demonstrate how the presented workflow can be used to efficiently retrieve, explore, and structure the large-scale, annotated production data required for a comprehensive investigation of such developmental phenomena.

5. Conclusion and Future Work

We have presented an integrated and reproducible pipeline for converting CHILDES data into a deeply annotated, queryable resource suitable for modern computational linguistics research. By combining robust CHAT parsing, standard NLP tools, and a powerful dependency query layer, our workflow automates a complex and error-prone process, thereby lowering the barrier to entry for quantitative studies of language acquisition.

However, we acknowledge several limitations. A

primary constraint is the performance of the external NLP tools themselves. The accuracy of the final linguistic analysis depends on the models provided by TreeTagger and UDPipe, which are not specifically trained on child language or child-directed speech. For the French data used in our case study, we found the TreeTagger parameters for spoken French trained on the data of the PERCEO project (Abeillé and Clément, 2006) combined with the UDPipe model 'French-GSD' to be reasonably effective, but performance may vary for other languages or more idiosyncratic speech patterns, especially verbless utterances (see Sagae et al. (2007) for a description of the specific challenges of parsing CHILDES data). Nevertheless, at least for part of speech, the pipeline offers a method for internal validation by including the tags from both the tagger and the parser, enabling the users to cross-check word-level annotations.

The scripts themselves are research tools and presume users comfortable with the command line (as well as with Python, if they wish to use different NLP tools); they are not packaged with a graphical user interface. While the CHAT parser is robust, it has been validated primarily on French, German, Italian, and North American English data from the CHILDES database; it may not yet handle all format idiosyncrasies present across the full, multilingual diversity of the TalkBank resources. Furthermore, the reliance on an external API for parsing introduces a dependency on network availability and the specific models offered by the service.

While the version described in this paper will be published as a release, future work will proceed in three main directions. First, we plan to extend the language-specific components of `childes.py` to provide robust support for more languages. Second, we aim to integrate parsers and their models in a containerised environment (e.g., via Docker) in order to remove the reliance on the external API, improve processing speed and grant the users more options. Finally, we intend to develop a library of Grew queries for a wider range of constructions relevant to developmental research.

6. Acknowledgements

Work presented here was funded by the Deutsche Forschungsgemeinschaft (DFG) grant no. 437487447: *Change and acquisition of verbal structures*, as part of the Research Unit FOR 5157 "Structuring the input in language processing, acquisition and change" (SILPAC).

I thank Thomas Rainsford, Mathilde Regnault, and three anonymous reviewers for their insightful comments and suggestions.

7. Bibliographical References

Anne Abeillé and Lionel Clément. 2006. Manuel d'annotation pour les corpus du projet PERCEO. https://repository.ortolang.fr/api/content/perceo-/1/Manuel-annotation.pdf.

Anna Cardinaletti, Sara Cerutti, and Francesca Volpato. 2021. On the acquisition of third person dative clitic pronouns in Italian. *Lingue e linguaggio*, (2):311–341.

Hélène Delage, Stephanie Durrleman, and Ulrich H. Frauenfelder. 2016. Disentangling sources of difficulty associated with the acquisition of accusative clitics in French. *Lingua*, 180:1–24.

Gaël Guibon, Marine Courtin, Kim Gerdes, and Bruno Guillaume. 2020. When Collaborative Treebank Curation Meets Graph Grammars. In *LREC 2020 - 12th Language Resources and Evaluation Conference*, Marseille, France.

Bruno Guillaume. 2019. Graph matching for corpora exploration. In *JLC 2019 - 10èmes Journées Internationales de La Linguistique de Corpus*, Grenoble, France.

Brian MacWhinney. 2000. *The Childes Project. 1: Transcription Format and Programs*. Erlbaum, Mahwah.

Brian MacWhinney. 2019. CHAT Manual.

Mihaela Pirvulescu and Nelleke Strik. 2014. The acquisition of object clitic features in French: A comprehension study. *Lingua*, 144:58–71.

Beth Randall, Anthony Kroch, and Beatrice Santorini. 2004. CorpusSearch 2 users guide.

Kenji Sagae, Eric Davis, Alon Lavie, Brian MacWhinney, and Shuly Wintner. 2007. High-accuracy annotation and parsing of CHILDES transcripts. In *Proceedings of the Workshop on Cognitive Aspects of Computational Language Acquisition*, pages 25–32, Prague, Czech Republic. Association for Computational Linguistics.

Helmut Schmid. 1994. Probabilistic Part-of-Speech Tagging using Decision Trees. Stuttgart. Institut für maschinelle Sprachverarbeitung.

Milan Straka. 2018. UDPipe 2.0 prototype at CoNLL 2018 UD shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium. Association for Computational Linguistics.

8. Language Resource References

Bassano, Dominique. 2005. *CHILDES French Pauline Corpus*. TalkBank. PID https://doi.org/10.21415/T5VS3W.

Champaud, Christian. 2004. *CHILDES French Champaud Corpus*. TalkBank. PID https://doi.org/10.21415/T5M88F.

Guillaume, Bruno. 2025. *Grew. Graph Rewriting for NLP*. PID https://grew.fr.

Jisa, Harriet. 2004. *PhonBank French Lyon Corpus*. TalkBank. PID https://doi.org/10.21415/T5M02D.

Le Normand, Marie-Thérèse. 2014. *CHILDES French MTLN Corpus*. TalkBank. PID https://doi.org/10.21415/T58S3M.

Leveillé, Madeleine. 2004. *CHILDES French Leveillé Corpus*. TalkBank. PID https://doi.org/10.21415/T5RK5N.

Morgenstern, Aliyah and Parisse, Christophe. 2009. *PhonBank French Paris Corpus*. TalkBank. PID https://doi.org/10.21415/T5PS3B.

Palasis, Katerina. 2020. *CHILDES French Palasis Corpus*. TalkBank. PID https://doi.org/10.21415/T5SW4P.

Plunkett, Bernadette. 2004. *CHILDES French York Corpus*. TalkBank. PID https://doi.org/10.21415/T5NK63.

Rasetti, Lucienne. 2004. *CHILDES French Geneva Corpus*. TalkBank. PID https://doi.org/10.21415/T5F898.

Schmid, Helmut. *TreeTaggger*. PID https://cis.uni-muenchen.de/ schmid/tools/TreeTagger/.

Stein, Achim. 2026. *French CHILDES Pipeline, version 3.0*. GitHub. PID https://github.com/michaniets/french-childes.

Straka, Milan. 2023. *UDPipe, version 2.1.0*. PID https://github.com/ufal/udpipe.

Vion, Monique. 2004. *CHILDES French Vion/Colas Corpus*. TalkBank. PID https://doi.org/10.21415/T5N300.

Yamaguchi, Naomi. 2015. *PhonBank French Yamaguchi Corpus*. TalkBank. PID https://doi.org/10.21415/T5ZP45.