

BACHELOR'S THESIS

---

**chatconllu: a Two-way Converter  
between CHILDES and UD Annotations**

---

BY

Jingwen Li

SUPERVISOR:

Dr. Çağrı Çöltekin

Seminar für Sprachwissenschaft  
Philosophische Fakultät

October 2021

# Abstract

Both *Child Language Data Exchange System* (CHILDES) and *Universal Dependencies* (UD) are projects designed to facilitate linguistic research with computerised tools and resources. Each project is paired with a standardised annotation format, which removes barriers to linguistic data-sharing introduced by diverse annotation or transcription habits.

The CHILDES *CHAT* and UD *CoNLL-U* formats, although widely accepted and used in their dominant fields, use different structures and tagsets to represent morphosyntax. With the help of a two-way converter, information can be augmented or cross-validated using existing tools developed for these two annotation formalisms.

To improve the interoperability between CoNLL-U and CHAT, a deterministic mapping for morphosyntactic information from CHAT codes to UD-style tags is created, and *chatconllu* is implemented as a command-line-based two-way converter. Within the limits of a Bachelor thesis, *chatconllu* is developed for and tested on several selected CHILDES databases of English, French and Italian, and the converted files are validated using CHILDES- and UD-maintained tools.

# Acknowledgements

People often overestimate the effort they put into their work and underestimate the many forces at play behind it. I would like to acknowledge all the *fateful coincidences* in my life that brought me here and produced this work. I would also like to give special thanks to my parents, especially my mother, who has been my best friend. I also want to thank Çağrı, my thesis advisor, both for his interest that started this project and his continued support, always giving detailed answers to my questions and providing me with helpful feedback.

# Contents

CHAPTER 1	<b>Introduction</b>	<b>1</b>
1.1	Related Work	2
CHAPTER 2	<b>CHILDES CHAT and UD CoNLL-U</b>	<b>3</b>
2.1	CHILDES CHAT Transcription System	3
2.2	UD CoNLL-U Annotation Scheme	5
2.3	Exclusive Information vs. Shared Information	6
CHAPTER 3	<b>Implementation of chatconllu</b>	<b>8</b>
3.1	Data Structures	8
3.2	Overview of chatconllu	9
3.3	CHAT to CoNLL-U	9
3.4	CoNLL-U to CHAT	14
CHAPTER 4	<b>Deterministic Mapping for Morphosyntax</b>	<b>16</b>
4.1	Part-of-Speech Tags	16
4.2	Morphological Features	18
4.3	Dependency Relations	19
CHAPTER 5	<b>Validation and Comparison</b>	<b>23</b>
5.1	Validating CHAT using CLAN Tools	23
5.2	Validating CoNLL-U using UD Tools	23
5.3	Comparison with Original Files	23
CHAPTER 6	<b>Conclusion</b>	<b>24</b>
APPENDIX A	<b>Morphosyntactic Mappings</b>	<b>25</b>
A.1	Part-of-Speech Mapping	25
A.2	Morphological Features Mapping	29
A.3	Dependency Relation Mapping	30
	<b>References</b>	<b>31</b>
	<b>Declaration of Authorship</b>	<b>33</b>

# 1 | Introduction

Traditionally, linguistic research has primarily been language-specific, with theories stem from studies on Indo-European languages. In recent decades, there has been much progress in crosslinguistic research, one of which is the *Universal Dependencies* (UD) project (Nivre et al., 2016 and Nivre et al., 2020).

As Joakim Nivre explained in his chapter *Towards a Universal Grammar in Natural Language Processing*, instead of applying a descriptive and explanatory account of morphosyntax, UD's universal grammar is developed for natural language processing applications (Nivre, 2015). Backed by both universal and language-specific tagsets, UD's *CoNLL-U* annotation allows for crosslinguistic corpus sharing. However, most UD treebanks are still based on written text.

Like CoNLL-U, *CHAT* is a standardised annotation scheme. It is part of the *Child Language Data Exchange System* (CHILDES) project (Macwhinney, 2000), and it offers a standard way to transcribe spontaneous speech. Morphosyntactic information can also be added as dependent tiers to the main utterances, and if preferred, it can be generated automatically by running *MOR* (Macwhinney, 2000), *POST* (Parisse et al., 2000) and *GRASP* (Sagae et al., 2004b).

It would be beneficial to use the advantages of both formalisms to compensate each other—CHILDES can provide UD with spontaneous speech treebanks, and UD can offer its tagsets which are widely accepted. To achieve this goal, *chatconllu*, a two-way converter, is designed and implemented as a command-line tool in this project.

This thesis project explores and experiments with format conversion between the above-mentioned annotation schemes via implementing a two-way converter in Python. The scope of analysis is limited to a selected few corpora from the CHILDES database, listed in Table 4.3. Transcriptions in CHAT format are first converted to CoNLL-U format and then converted back to see which information is lost in the process. With the aid of tools like *UDPipe* (Straka et al., 2016), new information is augmented to the corpora and, after *chatconllu*'s conversion, is represented in CHAT-style dependent tiers. Validation tests are performed using CHILDES- and UD-maintained tools to verify the well-formedness of the resulting files. This thesis also serves as a complement to the code on GitHub.<sup>1</sup>

1: [childes-ud/chatconllu: https://github.com/Mentha7/childes-ud](https://github.com/Mentha7/childes-ud)

The remaining part of the thesis is structured as follows: Chapter 2 gives necessary background information of both annotation schemes. Chapter 3 introduces the converter and its implementation. Chapter 4 discusses the mappings from CHILDES to UD. Chapter 5 explains the validation process and results. Finally, with Chapter 6, I conclude this thesis and propose future improvements for chatconllu.

## 1.1 Related Work

**Dependency Parsing for Low-resource Spontaneous Speech** Liu et al. used a low-resource setting, focusing on one child from the Brown Corpus (Brown, 1973) and tested the performance of dependency parsers on parent-child conversations (Liu et al., 2021). The out-of-domain parser was trained on written text and performed well on adult speech but not so well on child speech, while the parser trained on a limited amount of in-domain spoken data improved the parsing result of child speech to be comparable with adult speech. As part of their study, they created a semi-automatic adaptation of annotation formats from CHILDES to UD.

**The AnnCor CHILDES Treebank** Odijk et al. preprocessed the Dutch CHILDES corpora, augmented them with syntactic annotations using the Alpino parser (Bouma et al., 2000), and performed partial manual verification (Odijk et al., 2018). While creating this treebank, they noticed the many disadvantages of annotation discrepancies and argued that annotation guidelines should be stated explicitly and strictly followed to produce consistency. They suggest combining human annotations with automatic checks by computer programs. They also pointed out that although the grammar used in child speech is limited since the child is still acquiring the language, assigning adult syntactic structures to their utterances is still valid.

## 2 | CHILDES *CHAT* and UD *CoNLL-U*

This chapter introduces the two annotation schemes of interest—*CHAT* and *CoNLL-U*—in detail and gives the reader an idea of what information is shared by both annotations and what information is exclusively relevant to only one of them. This will shed light on the implementation of *chatconllu* which will be discussed in Chapter 3.

### 2.1 CHILDES *CHAT* Transcription System

#### 2.1.1 Motivation

The *Codes for Human Analysis of Transcripts* (*CHAT*) transcription system is developed for the *Child Language Data Exchange System* (*CHILDES*) project (Macwhinney, 2000) to aid child language acquisition research.

As MacWhinney pointed out in the *CHAT* manual, collecting audio data for spontaneous conversational interactions is easy, one just have to turn on the recorder, however, processing the collected recordings and turning them into usable data for linguistic research is a demanding task (Macwhinney, 2000).

For studies that rely on speech corpora, transcribing the conversations is an indispensable preprocessing step. Consistent treatment of raw data is critical to scientific research. Therefore, it is hard to imagine how difficult it was to maintain transcription standards when there was no standard format, especially for child speech, which is more chaotic than average adult speech data. Thus, *CHAT* is designed to accommodate the need for a standard transcription system of conversational speech. Moreover, *CHAT* transcripts can be analysed by TalkBank programs<sup>1</sup> like *Computerized Language Analysis* (commonly referred to as the *CLAN* Program), which facilitates the transcription, sharing and analysis of human conversational interactions.

1: <https://talkbank.org/software/>

#### 2.1.2 File Structure

For a *CHAT* file to be well-formed, there are some basic structural requirements. The minimal structure of a well-formed *CHAT* file is described by *minCHAT* (Macwhinney, 2000, pp. 21-22).

First of all, each CHAT file should have at least two parts:

- ▶ *headers* preceded by the @ symbol that stores metadata like language and participants. Some headers are obligatory,<sup>2</sup> some are optional. Despite its name, headers can be added in the middle of a CHAT file to represent information like tape locations. *Header* is likely a misnomer, and they should be called *meta* as they generally preserve higher-level information and are not confined to file-initial positions.
- ▶ the *main line*—utterances led by the asterisk symbol \*, which are the actual sentences being said.

2: Obligatory headers include: the file-initial header @Begin, @Languages, @Participants, @ID and the file-final header @End.

Then, additional information can be *optionally* added as dependent tiers. Annotations of morphological, syntactical, and phonological information or general remarks of the transcriber's choice all go into separate dependent tiers of the main utterance. Dependent tiers are preceded by the percent symbol %.

### 2.1.3 Dependent Tiers

Since almost any type of information can be organised into separate dependent tiers and added to the relevant utterances, the use of dependent tiers demonstrates the inclusiveness and extensibility of the CHAT format. The CHAT manual defines a standard list of dependent tiers (Macwhinney, 2000, pp. 81-87). The tier names are mostly lowercase three-letter codes, but with the exceptions of tier names preceded by an x, like %xmor. With the active use of dependent tiers, researchers can adjust the level of specificity in the CHAT files to meet their individual needs. At the same time, since dependent tiers are optional, they do not interfere with other obligatory parts of the file.

A dependent tier can contain one of two types of information:

- ▶ utterance-level information
- ▶ word-level information

**utterance-level information** Utterance-level information usually appears as descriptive remarks. For example, the action tier %act that describe actions by the speakers is shown in the example below.<sup>3</sup>

3: Excerpt from Adam/040217.cha of the CHILDES database for the Brown Corpus (Brown, 1973)

**Listing 2.1:** Example of a dependent tier with utterance-level information.

```
*PAU: 0 .
%act: hits Adam
```

**word-level information** Word-level information is associated with words in the utterance. Word-based strings encoded with information are separated by the space character. The below example<sup>4</sup> shows two dependent tiers with word-level information.

4: from the same file with Listing 2.1



**Listing 2.2:** Example of dependent tiers with word-level information

```
*MOT: what happened ?
%mor: pro:int|what v|happen-PAST ?
%gra: 1|2|SUBJ 2|0|ROOT 3|2|PUNCT
```

The two most commonly seen tiers are %mor and %gra tiers, which can be generated automatically by the use of a series of CLAN programs. The %mor tier encodes morphological information, including affixes, the stem and part-of-speech category of a word, while the %gra tier keeps the dependency structure of the utterance and the grammatical relations between head and dependent words.

## 2.2 UD CoNLL-U Annotation Scheme

The CoNLL-U format, used in *Universal Dependencies 2.7* (Zeman et al., 2020) is modified from the CoNLL-X format (Buchholz et al., 2006). A file in the CoNLL-U format can be seen as a collection of sentences, where each sentence is followed by an empty line to separate the sentences.

Each CoNLL-U sentence can be divided into two parts:

- comment lines<sup>5</sup> start with the hash symbol # with obligatory fields sent\_id and text. sent\_id corresponds to a unique index in the conllu file and text the text of the sentence. A minimal example is shown below:

```
# sent_id = 1
# text = why ?
```

- token lines where each line represent one token or punctuation in the sentence.

**token representation** Each token<sup>6</sup> in the text should possess one line, and each line is tab-separated into 10 fields:<sup>7</sup>

1. ID: word index starting from 1 for each new sentence.<sup>8</sup>
2. FORM: word form or punctuation as indicated in the sentence comment.
3. LEMMA: lemma of the word
4. UPOS: one of the 17 Universal part-of-speech tags.<sup>9</sup>
5. XPOS: language-specific part-of-speech tags
6. FEATS: a list of morphological features<sup>10</sup> separated by |.
7. HEAD: index of the head word of the current token, or 0 if the current token is the root.
8. DEPREL: one of the 37 universal syntactic relations in UD,<sup>11</sup> or one of its subtypes.
9. DEPS: head-deprel pair in the form of head:deprel.
10. MISC: any other information related to the current token.

5: Meta-information about the entire document is stored in the comment lines of the first sentence.

6: with the exception of multi-word tokens

7: For more information visit: <https://universaldependencies.org/format.html>

8: With the exception of empty nodes, which can be a decimal number greater than 0.

9: The UPOS tags are listed in Table A.1 in Appendix A.

10: They have to be from UD's universal feature inventory or from a predefined language-specific set accepted by UD.

11: The 37 universal syntactic relations can be found on: <https://universaldependencies.org/u/dep/index.html>

**additional constraints** The fields cannot be empty, if any information<sup>12</sup> is not specified for a field, an underscore \_ is used in place as a placeholder.

12: with the exception of word index

The tokens in the previous minimal example can be represented as follows. The top line in bold is added to indicate the field type:

<b>ID</b>	<b>FORM</b>	<b>LEMMA</b>	<b>UPOS</b>	<b>XPOS</b>	<b>FEATS</b>	<b>HEAD</b>	<b>DEPREL</b>	<b>DEPS</b>	<b>MISC</b>
1	why	why	ADV	_	PronType=Int	0	root	_	_
2	?	?	PUNCT	_	_	1	punct	_	_

**multi-word tokens** Multi-word tokens are treated as word groups which span multiple lines. A clear example is given on the UD website and is reproduced here to illustrate the point.<sup>13</sup>

13:

<b>ID</b>	<b>FORM</b>	<b>LEMMA</b>
1-2	vámonos	_
1	vamos	ir
2	nos	nosotros
3-4	al	_
3	a	a
4	el	el
4	mar	mar

This example is taken from <https://universaldependencies.org/format.html>. Fields other than ID, FORM and LEMMA are omitted for simplicity.

## 2.3 Exclusive Information vs. Shared Information

Understanding how the same type of information is organised and represented in these two formats is crucial to format conversion.

As discussed above, both CHAT and CoNLL-U store meta-information as well as morphosyntactic information of tokens in the utterance. However, upon reading the manuals of both formalisms, it seems that CHAT encompasses a broader range of information types otherwise not present in CoNLL-U formats, like the required headers of a CHAT file. Therefore, although metadata in CHAT is meaningless for CoNLL-U, to reproduce CHAT from a converted file in CoNLL-U format, it must be stored in such a way that it can be put back to the right place.

Since CHAT is more encompassing than CoNLL-U and CoNLL-U files have no exclusive information that requires previously non-existent tiers or headers, data contained in CoNLL-U files can be seen as a subset of that of the CHAT files.

The general information types contained in CHAT files are summarised in Table 2.1. It is clear from the table that file- and sentence-level metadata distributed in diverse places of a CHAT file are all stored as sentence comments in CoNLL-U.

**Table 2.1:** Information organisation in CHAT and CoNLL-U files

Level	CHAT			CoNLL-U
	Data type	Information	Storage	
<b>File</b>	Meta	obligatory headers	file begin and end	sentence comments
		other headers	middle of the file	sentence comments
<b>Sentence</b>	Meta	speech codes	main line	sentence comments
		speaker	main line	sentence comments
		dependent tiers	dependent tiers	sentence comments
		linkage to media	main line	sentence comments
<b>Token</b>	Text	sentence text	main line	sentence comments
	Morphological	word form	main line	FORM
		lemma	MOR stem	LEMMA
		part-of-speech	MOR pos	UPOS and XPOS
		features	MOR affixes	FEATS
	Syntactic	word index	GRA index	ID
		head	GRA head	HEAD
		syntactic relation	GRA GR label	DEPREL
	Additional	clitic type	MOR	MISC
		compound components	MOR	MISC
		special forms	main line	MISC
		...	...	MISC

As for the shared morphosyntactic information, while CHAT uses dependent tiers parallel to the main line, UD takes a token-based approach. Nevertheless, both formalisms use the *word* as the basic unit for morphosyntactic analyses. Thus, although placed in separate tiers, morphological information in the %mor tier and syntactic relations between words in the %gra tier can be associated with tokens in the utterance. Similarly, by combining information separated in different fields of the CoNLL-U token line, one can group and reorganise the fields to produce string representations similar to those assigned by MOR and GRASP.

## 3 | Implementation of chatconllu

### 3.1 Data Structures

From Chapter 2 we know that CoNLL-U files are collections of sentences, where each sentence has comment lines and token lines. With Table 2.1, I also showed that to keep as much information from CHAT files as possible, information exclusively relevant to CHAT had to be stored in sentence comments.

chatconllu takes the idea from the CoNLL-U sentence and proposes to use two object classes—Sentence and Token—to organise the diverse types of information in CHAT files. The benefit of using these two uncomplicated structures is that they correspond well to the CoNLL-U way of information organisation, making writing to CoNLL-U strings straightforward.

**Sentence** Each Sentence object is comprised of a list of Token objects and several additional attributes to store sentence-level information like dependent tiers and file-level metadata like headers preceding the current utterance. The list of Token objects corresponds to the words in the normalised utterance. A normalised utterance is an utterance with the CHAT transcription codes removed. When a MOR tier is present, the tokens should have a one-to-one correspondence with the strings in the MOR tier.

**Token** The attributes of the Token object include the 10 fields of a normal CoNLL-U token, with two additional attributes:

- ▶ `multi`—to store the end index of a multi-word token.
- ▶ `type`—to store the clitics type

Since Token is organised parallel to the UD-style token, it can be directly converted to a token line.<sup>1</sup>

<sup>1</sup>: Or lines for multi-word tokens

## 3.2 Overview of chatconllu

The behaviour of chatconllu can be described in plain English: For all files or files with name specifications in the given directory, depending on the input file format, convert them to the other format and save the output as different files. I used the *pyconll*<sup>2</sup> package developed by (Matías Grioni, 2018) for parsing the converted CoNLL-U files and accessing the values of token fields. I chose to use it instead of writing my own script because I want to quickly validate the general structure of the converted files. Using the load method from a distributed package allows me to get instant feedback if the file is malformed. chatconllu uses argparse to process inputs from the command line. With optional arguments,<sup>3</sup> it is also possible to finetune the desired output by asking chatconllu to generate new dependent tiers or omit existing dependent tiers during format conversion.

2: pyconll is a low-level API for processing CoNLL-U formatted files.

3: All commands of chatconllu can be found via the chatconllu --help command after proper installation.

## 3.3 CHAT to CoNLL-U

The conversion from CHAT to CoNLL-U is outlined as follows:

1. parse the CHAT file to obtain utterances grouped with dependent tiers
2. clean the utterance by removing CHAT transcription codes
3. extract token information from the utterance-tiers group to construct Token objects
4. construct Sentence object with the created Token objects and other utterance-level information
5. write the sentences in CoNLL-U format to an output file

### 3.3.1 Normalising Utterances

To clean the utterances, I wrote a helper method `normalise_utterances`. This method is a simple parser by itself. The input of this method is an utterance with CHAT transcription codes. The output is a tuple where the first value is a list of tokens in the utterance. After the removal of transcription codes, if the %mor is given, the length of the list should match the number of strings in the %mor tier. The second value is simply the line itself, which is to be stored as-is as a sentence comment in the output file, allowing for the recovery of the original utterance in the back-conversion process.

**regular expressions** To remove the CHAT transcription codes, one has to first identify them in the utterance string. Regular expressions are designed for the codes described in the CHAT manual and are tested on the chosen corpora. It is worth noticing that although some regular expressions can be merged into one, they are deliberately left not to, to keep them manageable for future human inspection, addition, editing, and case-wise debugging.

**nested angle brackets** The original algorithm I used was simply parsing the string by incrementing the index of the current character and matching the remaining line with defined regular expressions. It was successful in processing flat structures, but not nested structures. Although nested structures are arguably rare in the corpora and the max level of nesting is just one, I still wanted a better solution to the problem.

In CHAT, nested structures occur with *scoped symbols* (Macwhinney, 2000, Part 1, p.72). A scoped symbol is a marked segment of speech. The tokens in the segment are enclosed in angle brackets, while the markers which describe the segment are enclosed in the square brackets that follow. There can be more than one square-bracket-enclosed descriptor following an angle-bracketed speech segment, but no other materials are allowed between the scope enclosed in angle brackets and the symbols enclosed in square brackets.

The challenge of nested angle brackets can be illustrated using two example utterances:<sup>4</sup>

**Listing 3.1:** Example utterances with nested angle brackets

```
<here <monk(ey)> [/] monkey> [>]}
<<yeah yeah> [/] yeah> [?]}
```

4: These examples are taken from the CHILDES database of the Brown Corpus (Brown, 1973).

To better understand the examples in Listing 3.1, I also introduce two operations triggered by the markers:

- ▶ *deletion*: the previous word or scoped element, as well as this marker, should be removed, triggered by markers like [/].
- ▶ *keep*: keep the previous word or the words in the scoped element, remove this marker after that, triggered by markers like [?] and [>]. Note that the > character also appears as a descriptor, making things more complicated.

With these two operations defined, it soon becomes clear to the human eye that the two examples should be normalised to *here monkey* and *yeah* after being cleaned. However, the same parses can only be achieved by a computer program using a stack. A stack is used to remember the number of opening angle brackets that have been seen so that the current nesting level is known. With the help of a stack, the first > is correctly associated with the second < in the first sentence and the first < in the second example.

In chatconllu, nested expressions like the examples above are translated into nested lists with the help of a depth counter and a push operation. The depth counter is like a stack and keeps the current level of nesting, and the push operation pushes the word to the list of the current depth. The algorithm for processing nested structures is listed in Algorithm 1. After the utterance string is processed by the `normalise` method, a nested list corresponding to the structure of the scopes in the utterance is returned. Then, a recursive delete method is used to start removing unwanted elements from the innermost list. Then, the list is flattened and scanned for other operations like `omit`. The final remaining list is then the list of all tokens to keep.

**Algorithm 1:** Dealing with nested angle brackets**Function** Push(*object*, *list*, *depth*):

```

    if depth < 0 then                                /* mismatch */
        raise error
    while depth > 0 do
        list ← last element of list /* go one level deeper */
        depth ← depth − 1
    list.append(object)

```

**Function** Normalise(*line*):

```

    i ← 0
    depth ← 0
    tokens ← []
    chars ← []
    while i < len(line) do
        if line[i] == "<" then
            push([], tokens, depth)
            depth ← depth + 1          /* stack push */
        else if line[i] == ">" then
            token ← join(chars)
            chars ← []
            if token then
                push(token, tokens, depth)
            depth ← depth − 1          /* stack pop */
        ...
        else                                /* read normal character */
            chars ← line[i]
        if i == len(line) then              /* add final token */
            token ← join(chars)
            chars ← []
            if token then
                push(token, tokens, depth)
            i ← i + 1
    return tokens

```

**to normalise or not** Although chatconllu can clean the CHAT annotation codes relatively well, it remains a question whether one really wants to obtain the *clean* version of the utterance. On the one hand, word-based information in %mor and %gra tiers is only associated with tokens in the cleaned utterance. Therefore, to use the information provided in these tiers, one has to remove unannotated words (or nonwords). On the other hand, removing things like speech errors makes spontaneous conversational data less speech-like. Or worse, it could make the real sentence structure of the sometimes already ungrammatical utterance even less obvious, at least for dependency parsers. In the study of creating a dependency treebank using Dutch CHILDES corpora, during the process

of cleaning the encoded utterances, Odijk et al. observed that blindly removing unintelligible words results in a decrease in the performance of the dependency parser they used (Odijk et al., 2018). For example, although xxx is a nonword, it still functions as a placeholder.<sup>5</sup> The Alpino parser (Bouma et al., 2000) they used seems to depend on the positional information, despite the word is unknown. This was again confirmed in a way by (Liu et al., 2021), where they found utterances with omissions are more undesirable in dependency parsing and affected the performance of the dependency parser. Therefore when it comes to preprocessing spoken data like the CHILDES corpora. The degree of normalisation depends on the intended use case.

5: Think of it as 0 in digits or as a masked token.

### 3.3.2 Extracting Information from Dependent Tiers

Since morphosyntactic information in %mor and %gra is kept with individual tokens in CoNLL-U files, they have to be processed in such a way that all relevant information from both tiers (if present) are associated with the respective token in the utterance. It can be more complicated with clitics and contractions, which results in different lengths of the %mor and %gra tiers, shown in Listing 3.2.<sup>6</sup> Clitics, like expressions involving English auxiliary verb 've, the possessive 's, and contractions, like the English negation particle *n't* or the French *articles composés-du*,<sup>7</sup> are treated by the MOR grammar as one word group and assigned a single string with MOR codes, while GRASP gives a separate string representation for each of its parts.

6: The examples in Listing 3.2 are taken from the same file with Listing 2.1

7: "du" is the contraction of "de" and "le"

**Listing 3.2:** Example to show the different number of strings in %mor and %gra tiers due to contracted form *that's*.

```
*CHI: dat's [: that's] hard .
%mor: pro:dem|that~cop|be&3S adv|hard .
%gra: 1|2|SUBJ 2|0|ROOT 3|2|JCT 4|2|PUNCT
```

#### 3.3.2.1 Morphological information from %mor

The syntax of the MOR annotations is complicated by the existence of *word groups*. Word groups are the root cause of the sometimes inconsistent number of annotation strings between %mor and %gra. Since the strings in the %mor tier have a strict one-to-one correspondence with words on the main line, annotations of clitics and contractions like *that's* also have to be single strings.

Here, I introduce the syntax of the MOR annotations for words and word groups, which include clitics and compounds.

**word** The core structure of an individual word is `pos|stem`, giving only the part-of-speech and stem of the word, but words ususally receive more complex morphological analysis, following the structure below:

```
prefix#pos|stem&fusionalsuffix-suffix=translation
```



A word can have any number of prefixes, each followed by a hash #, fusional suffixes, each preceded by an ampersand & and suffixes, each preceded by a dash - (Macwhinney, 2000, Part 3, p.7).

**word group** A word group has the word annotation of each word in the word group joined by one of three types of delimiter: the dollar sign \$ for preclitics, tilde ~ for postclitics, and the plus sign + for compounds. Compounds, however, has not only each component word annotation preceded by +, but also receives an additional POS tag which is attached to the beginning of the MOR annotation for the word group. For example, *peanut+butter* in the main line has the MOR annotation `n|+n|peanut+n|butter`.

Morphological information encoded in the %mor tier is listed in Table 3.1. The part-of-speech and features can be extracted directly by parsing the MOR annotation string. They are converted to UD-style values using a deterministic mapping which will be introduced in the next chapter. During back-conversion from CoNLL-U format, chatconllu keeps the original MOR POS tags as language-specific tags in XPOS to reconstruct the MOR annotation. In the future, they should be moved to the MISC field as well. Other information requires more processing and organisation, which I explain below:

**lemma** The syntax of the MOR annotations shows that getting the lemma out of a MOR string is not as trivial as it seems. There are several pitfalls to avoid:

- ▶ The semantics of the symbols may be ambiguous. Take the dash symbol - as an example, for words like *tic-tac-toe*, the - merely connect parts of the word, but in the example of `part|get-PRESP` for the word *getting*, - could be an indicator for the present participle suffix -ing.
- ▶ One should also notice that the stem is not always the lemma, just that for English, these two share the same form in most cases. For instance, for *untied*, we have the MOR segment `un#v|tie-PASTP`, which means in plain English that the word has a stem *tie* with prefix *un*, its part-of-speech is verb and this word form is the past participle of the verb. Since in CoNLL-U, the LEMMA field is reserved for lemma and not word stem, the prefix should be put back in place to produce the real lemma.
- ▶ Compound strings are structured differently and cannot be parsed in the same way. For words like `n|+n|peanut+n|butter`, instead of parsing this string using a different method, chatconllu simply use the word form as lemma. In this example, the word form and the lemma should both be *peanutbutter*.

**misc** The extensibility of CHAT enables transcribers to note down almost any information they want, but the CoNLL-U format is focused on selective grammatical information of the tokens in the sentences and only values defined either universally or language-specifically by UD

**Table 3.1:** Information contained in MOR strings that needs to be extracted, and the corresponding CoNLL-U field to store this information.

MOR	Field
lemma	LEMMA
part-of-speech	UPOS, XPOS
features	FEATS
clitic type	MISC
compound	MISC
translation	MISC

are accepted. To protect against the loss of information during format conversion, chatconllu uses the MISC field to store information that is outside of the previous nine fields. One should also take care not to use reserved symbols like the field-value separator `|`. For instance, in one wants to save the compound components `+n|peanut+n|butter` for `n|+n|peanut+n|butter` as is in , the `|` character should first be replaced with semantically insignificant symbols like `@`, and only to be replaced back to `|` after the MISC field is processed as a dictionary during the reconstruction of dependent tiers.

### 3.3.2.2 syntactic information from %gra

The grammatical relation (GR) annotation in %gra is represented in a simpler form than the annotation in %mor, as shown in Table 3.2. And since the annotations in GRA are not affected in form by compounds or multi-word tokens, they can be parsed uniformly. Therefore, I only explain the GRA annotation syntax here for completeness.

A triple, (dependent, head, relation type), is used to denote a dependency relation. In the %gra tier, the annotation of such an relation is of the form `i|j|g`, where:

- ▶ `i` is the index of the current word, which is the dependent
- ▶ `j` is the index of the head word
- ▶ `g` is the GR label used for this relation

**Table 3.2:** Information contained in GRA strings that needs to be extracted, and the corresponding CoNLL-U field to store this information.

GRA	Field
word index	ID
head	HEAD
syntactic relation	DEPREL

### 3.3.3 Sentence-level Information

Sentence-level info can also be stored in the comments in CoNLL-U. Meta-information given in the headers of CHAT files are saved as file-initial comments before the first sentence because they are file-specific. However, one has to admit that storing information like this reduces readability. Another thing is that information stored in this way are susceptible to changes accidentally made, which cannot be detected by format validators. One possible solution is to validate the respective format before and after conversion is performed.

## 3.4 CoNLL-U to CHAT

The back-conversion from CoNLL-U to CHAT is more straightforward than from CHAT to CoNLL-U because CoNLL-U format has a clear structure of organising information. All file- and sentence-level meta-data are stored in the comments, and all morphosyntactic information is already categorised and organised into the ten fields of UD token representation discussed in Section 2.2. I describe the processes involved below.

First, with the help of pyconll (Matías Grioni, 2018), chatconllu loads the CoNLL-U file to obtain pyconll sentences, each of which is composed of comments and tokens. The comments are read as a dictionary with

the field names like `sent_id` and `text` being the keys. Then `chatconllu` opens an output file, extract and write information in the same order as it would appear in a CHAT file, first headers, then the utterances and dependent tiers and finally the last utterances with no tokens and the file-final headers. During the process of writing data to the output file, if `%gra` or `%mor` is detected, information is reorganised using the syntax of the respective dependent tiers explained earlier. The user can optionally ask `chatconllu` to generate new empty `%gra` or `%mor` tiers, with the unspecified values being replaced by an underscore `_`. If the converted CoNLL-U files are processed by other tools like UDPipe (Straka et al., 2016) to add or change information, the user can choose to generate new tiers to represent the added information. Both or one of the following new tiers can be added by `chatconllu`:

- ▶ `%pos`—mirrors the `%mor` tier in CHAT but without converting back UPOS tags to the POS tags used by MO R. No morphemes are identified by the UD features as well.
- ▶ `%cnl`—corresponds to the `%gra` tier. Information also organised as a triplet string of the form `id|head|deprel`, without converting the values back to GRA codes.

## 4 | Deterministic Mapping for Morphosyntax

The previous chapter talked about how CHAT and CoNLL-U use MOR and GRA codes to structurally represent morphosyntactic information. However, the set of tags used in the two annotation schemes could also be semantically different. This chapter presents a deterministic mapping from CHAT codes to UD-style labels. The mapping consists of three major categories: part-of-speech tags, morphological features and dependency relations and is created for English, French and Italian.

### 4.1 Part-of-Speech Tags

The CHAT manual lists 39 POS tags for English (Macwhinney, 2000, Part 3, p.8). This set can be extended with language-specific tags for other languages. UD, on the other hand, uses a universal POS (UPOS) tagset with only 17 tags for all languages in the UD project.

UD's UPOS is developed based on Petrov's *A Universal Part-of-Speech Tagset*, where the crosslinguistic universality of the UPOS tags stems from their operational definitions instead of presupposing the existence of actual grammatical categories (Petrov et al., 2012). In CHAT, however, POS tags are much less universal across languages. Typologically different languages may use quite different tagsets than those prescribed by the MOR program for English (Macwhinney, 2000, Part3, p.7). Additionally, researchers can define their own POS tagsets, which not only means that, in terms of POS tags, CHILDES took an opposite approach from UD, but also makes an exhaustive mapping between POS tags used in the two annotation schemes impossible. Therefore, the mapping I define here concerns only the standard POS tagset given by the MOR grammar for English, French and Italian.

The adaptation from MOR POS categories to UD's UPOS can be achieved in the following ways:

- ▶ create a direct mapping between two tags if they share the same semantics or have a straightforward relationship
- ▶ collapse MOR POS categories into one UPOS tag when applicable

**Table 4.1:** Examples of MOR POS categories that can be directly converted to UPOS tags. See Table A.2, Table A.3, and Table A.4 in Appendix A for the full mapping.

MOR	UD
adj	ADJ
adv	ADV
coord	CCONJ
n	NOUN
num	NUM
pro	PRON
n:prop	PROPN
comp	SCONJ
v	VERB
det, qn	DET
co, fl	INTJ
dia, none	X
post, prep	ADP
aux, cop, mod	AUX
beg, end	PUNCT
inf, neg, part	PART

- make best-effort decisions based on the description of the POS categories by both formalisms
- map MOR POS categories with no UPOS counterparts to X

An example of a straightforward mapping between MOR POS codes and UPOS tags is shown in Table 4.1. A total of 16 out of the 17 UPOS tags are used in the tagset mapping, with the exception of *SYM*, which is reserved for symbols.

Collapsing of POS categories can be illustrated by the example of auxiliary verbs. The MOR code differentiates between auxiliary verbs *aux* or *v*: *aux*, copulas *cop* and modal verbs *mod*. However, since UD took an operational approach to define grammatical categories, it does not distinguish copulas or modal verbs from other verbs that function as auxiliaries. By doing so, UPOS can be applied to typologically distant languages.

The following MOR POS codes require more thought during the creation of a deterministic mapping, partially because they do not exist in all languages. For tags like these, UD usually raises them to a more coarse category and represents the exact type using morphological features instead.

**art** Articles are generally marked as *DET* in UD Treebanks with feature *PronType=Art*<sup>1</sup>.

**preart** *preart* are contractions of prepositions and articles, for instance the Italian *alla* is the contraction of *a* and *la*. In UD-style annotations, words like *alla* are treated as multi-word tokens, and a POS tag is given to each part of the token. However, CHAT considers them to be single words with forms of their own. To keep the alignment with the %mor tier, here *preart* is mapped to *DET* because the word functions as a determiner.

**vimp** Imperatives are tagged with *vimp* in %mor tier. However, in UD, imperatives are tagged with *VERB*, which is the category for all finite verb forms.<sup>2</sup> In UD, gender, person, number, tense, aspect, mood and voice are all specified as features. Therefore *vimp*, listed here as an example for finite verbs, is mapped to *VERB* with feature *MOOD=Imp*.

**vpfx** Preverb/verb particles are tagged with *vpfx*. Although they are particles, according to UD's definition,<sup>3</sup> they belong to *ADP*.

**conj** Despite the similarity between the MOR *conj* and UD *CCONJ*, they represent different things. UD distinguishes between *CCONJ*<sup>4</sup> and *SCONJ*<sup>5</sup>, the former is for coordinating conjunctions like *and* and *but*, which are tagged *coord* by MOR, and the latter marks subordinating conjunctions with complementizers like *that* or *if*, or adverbial clause introducers like *when*, these are tagged *conj* by MOR. Therefore, the suitable UPOS tag for the MOR POS tag *conj* is *SCONJ*.

1: Definition for *DET* can be found from <https://universaldependencies.org/docs/u/pos/DET.html>

2: Definition for *VERB* can be found from <https://universaldependencies.org/docs/u/pos/VERB.html>

3: Definition for *ADP* can be found from <https://universaldependencies.org/docs/u/pos/ADP.html>

4: Definition for *CCONJ* can be found from <https://universaldependencies.org/docs/u/pos/CCONJ.html>

5: Definition for *SCONJ* can be found from <https://universaldependencies.org/docs/u/pos/SCONJ.html>

**special form markers** It is worth noticing that some MOR codes do not correspond to any grammatical categories. For example, in CHAT, the so-called *special form markers* have unique POS tags independent of the part-of-speech category to which the word belongs (Macwhinney, 2000, Part 1, pp. 45-47). Nevertheless, some of these tags can still be mapped to UPOS. The tags *sing* for singing, *bab* for babbling and *wplay* for wordplay can all be mapped to INTJ, which stands for *interjection*. The definition of interjection, however, is vague (Ameka, 1992). UD defines interjection descriptively.<sup>6</sup> Following UD’s definition for INTJ, the core criteria for a word<sup>7</sup> to be an interjection include:

- ▶ that it is often used exclamatorily
- ▶ that it typically expresses emotions
- ▶ that it plays no syntactic role in the utterance
- ▶ that it might not be a recognised word in the language

Since singing, babbling and wordplay satisfy these criteria, along with the MOR code *co* for interjections, they are granted the INTJ tag. Other special form markers, on the other hand, have no corresponding UPOS tags whatsoever, for example *dia* for dialect forms. Unfortunately they have to be assigned to X, as words that does not belong to the other 16 categories.

**punctuation marks** Finally, MOR also defines a set of punctuation marks with string representations, shown in Table 4.2. Although treated as lexical items in the %mor tier, they are all given the PUNCT tag.

## 4.2 Morphological Features

Since MOR uses a morpheme-based approach to analyse individual words, morphological features like *tense* and *mood* are encoded by morpheme codes. Unlike MOR’s dissecting analysis of a word, UD takes a lexicalist view on morphology and treats word as a whole lexical item. Features are thus considered properties of the word. The existence of morphemes is not presupposed.

These two fundamentally different views of words are reflected in the different ways of structuring morphological information. CoNLL-U format keeps all relevant features of a word in the token’s FEATS field. And although each language in UD has its subset of morphological features, the feature types and values are by convention the same. Compared with UD, CHILDES’s way of handling features is less direct. Not all features are explicit as UD’s feature-value pairs. Instead, they have to be inferred from the morpheme codes or POS tags.

Listing of all possible MOR codes for grammatical morphemes is, again, unlikely. Therefore, I chose the easier path of collecting these codes from the CHILDES databases that I decided to work with, listed in Table 4.3.

6: "An interjection is a word that is used most often as an exclamation or part of an exclamation. It typically expresses an emotional reaction, is not syntactically related to other accompanying expressions, and may include a combination of sounds not otherwise found in the language."

Excerpt from <https://universaldependencies.org/docs/u/pos/SCONJ.html>

7: or non-word in this case

**Table 4.2:** MOR punctuation marks and their corresponding MOR codes.

Punctuations	MOR code
‡	beg
,	cm
”	end
“	bq
”	eq
‘	bq2
,	eq2

**Table 4.3:** CHILDES databases chosen for this project.

CHILDES database	Language
Belfast	eng
Brown	eng
English-MiamiBiling	eng
Wells	eng
Geneva	fra
Leveille	fra
MTLN	fra
vioncolas	fra
Tonelli	ita

Of all the morpheme codes found in these corpora, not all of them correspond to UD features. Since morphological features are not mandatory, I only created mappings for those with a corresponding UD feature-value pair. An example mapping is given in Table 4.4 on the right-hand side. Only 9 types of features can be obtained directly from the morpheme codes I encountered: Number, Person, Gender, Mood, Degree, Tense, Verbform, Voice, and Poss. All possible UD features and values allowed for English, French, and Italian can be found in Table A.5 in Appendix A. The codes are sometimes capitalised and sometimes not due to language- or corpus-specific conventions. They are all converted to lowercase for easy comparison. The original morpheme codes are kept in the MISC field.

### 4.3 Dependency Relations

In CHAT annotations, syntactic relations are represented by Grammatical Relations (GRs). They were first devised by Kenji Sagae (Sagae et al., 2004a) for the English language. After later revisions in 2007, CHAT now has 37 grammatical relation types (Sagae et al., 2007). They correspond to dependency relations stored in the DEPRELS field in CoNLL-U files. However, CHILDES and UD use different labels for these dependency relations and quite different parsing schemes. Sometimes, a conversion from GR to deprel also introduces a change in the structure of the dependency tree.

Defining a deterministic mapping between GRs and UD dependency relations thus ranges from direct label translation to linguistically-informed decisions. Table 4.5 gives examples of direct label conversions. The full mapping for the current version of chatconllu can be found in Table A.6.

A solid mapping requires both familiarity with the grammatical relations defined in the two formalisms and profound linguistic knowledge for the languages involved, this mapping I created is only a first attempt with many imprecisions. Therefore, instead of explaining the decision process with my limited linguistic knowledge, I choose to identify and discuss several hard-to-decide cases and cases where CHILDES and UD provide different dependency parses. Since only the English databases are annotated with syntactic relations, all labels and discussions here apply only to the English language.

The current version of chatconllu does not handle most of the cases discussed below, which is because it does not seek to produce gold-standard UD-style dependency parses. Instead, chatconllu only converts the files to CoNLL-U format, which can then be processed by other dependency parsers, for example, those trained on UD treebanks. Therefore, my best effort in designing this mapping goes to transforming existing GR labels to UD labels without changing the dependency structures produced by GRASP in most cases to aid comparison between the two parsing schemes.

**Table 4.4:** Example MOR grammatical morpheme codes and their corresponding UD feature-value pairs.

MOR	UD feats
3s	Number=Sing Person=3
cp	Degree=Cmp
f	Gender=Fem
inf	VerbForm=Inf
imp	Mood=Imp
impf	Tense=Imp
pass	Voice=Pass
pastp	Tense=Past VerbForm=Part
poss	Poss=Yes

**Table 4.5:** Example CHILDES grammatical relations and their corresponding UD dependency relations.

GRA	UD deprel
SUBJ	nsubj
CSUBJ	csbj
OBJ	obj
OBJ2	iobj
DET	det
APP	appos
AUX	aux
CONJ	conj
COORD	cc
PUNCT	punct
ROOT	root





Figure 4.1: CHILDES vs UD dependency graphs: interjections



Figure 4.2: CHILDES vs UD dependency graphs: vocatives

4.3.1 The Multiple-Root Problem

In UD, every sentence has exactly one root, represented by a single-root dependency tree. The dependency analysis generated by GRASP is, however, not always like that. After being processed by GRASP, sentences with non-final punctuations, like after an exclamation, a vocative expression or, close to the end of the utterance, before the tag question, receives multiple roots. Although usually only one ROOT or INCRROOT label is assigned, there can be multiple words whose head is marked 0, which, in effect, corrupts the single-root tree structure.

This problem occurs because exclamations and vocatives are not considered part of the clause. Tag markers are, however, for the chosen few corpora, not so much of a problem and usually have their head marked to the functional root (ROOT or INCRROOT), like in Figure 4.3.<sup>8</sup>

Consider the following sentences:

- (1) Sentences with non-final punctuations

a. [Well]<sub>INTJ</sub>, that's life. interjection

b. [Son]<sub>VOC</sub>, that's life. vocative

c. That's life , [isn't it]<sub>TAG</sub>? tag question

After running GRASP, sentences like (1) a. and (1) b. will be analysed like the left-hand side dependency graph of Figure 4.1 and Figure 4.2, while UD-style dependency analyses are shown on the right-hand side.

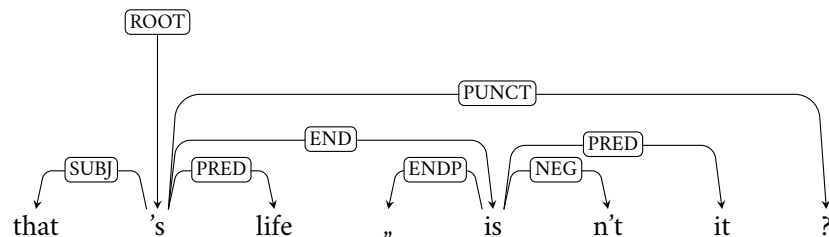


Figure 4.3: CHILDES: tag questions

8: Although from my observations, tag questions present a different problem of inconsistent parses of the same structure.



It is clear from the dependency graphs that the CHILDES dependency trees are broken. Therefore, to produce a proper dependency tree from that, one must change the head index of the token identified as the fake root (indicated by the GR label BEG) to that of the sentence’s true root. After the change-head operation is applied, the current token with GR label BEG is connected to the remaining single root. Now the two examples in Figure 4.1 and Figure 4.2 share the same dependency structures. However, dependency relation labels for vocatives and interjections should be different. To make the final decision, chatconllu needs the POS tag of the token with BEG. The decision process is outlined below:

- ▶ If the token has a POS tag and it is not INTJ, PROPN, or NOUN, the corresponding `deprel`<sup>9</sup> label should be `parataxis` for a loose side-by-side placement of the affected elements.
- ▶ If the POS tag is INTJ, then the first part of the sentence is considered as an exclamation. The corresponding UD-style label should be `discourse`.<sup>10</sup>
- ▶ If the POS tag is PROPN or NOUN, the first part of the sentence is considered as an expression in vocative. Therefore, the corresponding `deprel` label should be `vocative`.<sup>11</sup>

9: Definition for `parataxis` can be found from <https://universaldependencies.org/u/dep/parataxis.html>

10: Definition for `discourse` can be found from <https://universaldependencies.org/u/dep/discourse.html>

11: Definition for `vocative` can be found from <https://universaldependencies.org/u/dep/vocative.html>

### 4.3.2 Inconsistent Dependency Parses

When searching for cases of tag questions, I noticed that the same tag question is parsed differently by GRASP. Right now, chatconllu cannot change them into UD-style dependency structures automatically, so the structures, although problematic, are kept as they are. Only the dependency labels are naïvely changed to their UD counterparts according to the mapping I created.

#### 4.3.3 Predicate Relations

In GRASP’s syntactic analysis scheme, the grammatical relation `PRED` identifies a nominal or adjectival predicate and the predicate is seen as the argument of its head, which is a verb (Macwhinney, 2000, Part 3, p.63). However, in UD-style dependency annotations, predicate with copula verb constructions favours the predicate as the head, the differences can be seen from the two parses of the same sentence in Figure 4.4.

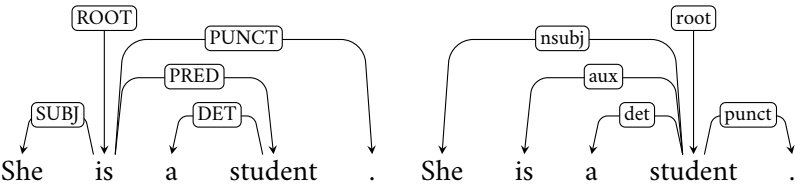
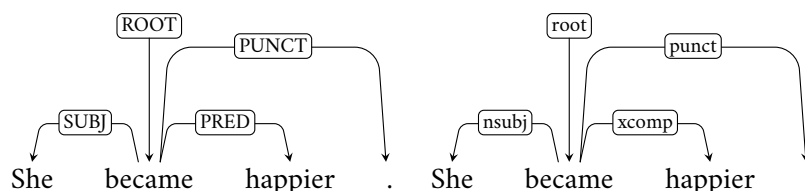


Figure 4.4: CHILDES vs UD: nominal predicate with copula verbs

However, constructions of predicates with linking verbs, like the example in Figure 4.5 with the verb *become*, receive the same dependency structure.



**Figure 4.5:** CHILDES vs UD: predicate with linking verb *become*

In fact, although CHILDES' analyses of copula structures are consistent, always making the copula verb the root, UD treats the cases differently. Three types of copula constructions that produce different UD-style parses are identified with comparison to the CHILDES parses in (Liu et al., 2021):

- ▶ copula structure with statements like the example in Figure 4.4
- ▶ copula structures in Wh- and How-questions:  
The interrogative words are analysed as the subject of the construction in CHILDES, while in UD-style analysis, the predicate becomes the subject.
- ▶ copula construction in expletives like *there be* expressions:  
CHILDES has *there* analysed as the subject, while in UD-style analysis, the predicate in *there-be* constructions is the subject.

This shows that to obtain UD dependency parses require not only a mapping of the labels used for the grammatical relations but also a case-by-case analysis with the annotation conventions of both formalisms in mind, which presents a problem for automatic conversion.

Many other cases that are challenging for adapting CHILDES syntactic annotations to UD-style are discussed and implemented in (Liu et al., 2021), which are not handled by chatconllu. They have made the code for conversion public on GitHub.<sup>12</sup>

12: [https://github.com/zoeyliu18/Parsing\\_Speech/tree/main/code](https://github.com/zoeyliu18/Parsing_Speech/tree/main/code)

Apart from that, Liu et al. observed that depending on the actual meaning of the child, the utterance *man no spoon* can be interpreted in at least three different ways (Liu et al., 2021). Odijk et al. also reported various reasons for a child to produce speech different from the conventional form used by adult speakers. Sometimes the true meanings are concealed by the exact transcriptions, which requires *intensive investigation of each phenomenon* by researchers (Odijk et al., 2018). It seems that for spoken data like child speech, information like context and tones that help us disambiguate meanings are not conveyed by literal transcriptions. Merely looking at utterance texts, it is even hard for a human to decide the real meaning of the utterance. Even if rules are defined for each grammatical relation, incomplete utterances can still produce unintended results in a decision making process. Therefore, instead of coming up with case-based rules to convert the syntactic relation labels and structures, chatconllu stores the original label in the token's MISC field and uses a direct mapping to convert the dependency label while keeping the tree structure produced by GRASP, except for sentences with multiple roots.

## 5 | Validation and Comparison

### 5.1 Validating CHAT using CLAN Tools

After using `chatconllu` to convert the CoNLL-U files back to CHAT format, the user may want to validate that the resulting CHAT files are well-formed and accepted by the CLAN program. CLAN has a CHECK program<sup>1</sup> that can validate the CHAT transcription format. Alternatively, if the user wants to stick to the command line, they can download Chatter,<sup>2</sup> a Java program from TalkBank, and follow the instructions on the website.

1: CLAN can be downloaded via <http://dali.talkbank.org/clang/>

2: Chatter - CHAT format validator: <https://talkbank.org/software/chatter.html>

For the chosen corpora, CHAT files produced by `chatconllu` pass the CHECK program in CLAN but not Chatter for the reason of missing tabs. Long lines in the original cha files generated from the CLAN program are broken into separate lines after a certain amount of characters is reached. Continuations of the line are preceded by an initial tab. However, the current version of `chatconllu` does not handle the maximum length of a line, and because of that, it fails the Chatter validation process.

### 5.2 Validating CoNLL-U using UD Tools

UD also provides tools for format validation. One of the UD maintained tools is the script `validate.py`.<sup>3</sup> Its validation is in five levels. To run the script, one has to input a language code because this script also checks whether the values in the fields are allowed for that language in high-level tests. Currently, `chatconllu` only passes the first two levels at best, which does not check if the values are accepted by UD or specifications of the given language.

3: link to file on GitHub: <https://github.com/UniversalDependencies/tools/blob/master/validate.py>

### 5.3 Comparison with Original Files

The differences between the original and back-converted cha files can be visually inspected using terminal tools like `icdiff`.<sup>4</sup> A side-by-side view of the two cha files selected is displayed in the terminal, with the differences coloured. As mentioned above, the most visible differences are the results of the missing leading tabs. Unfortunately, I could not figure out a way to solve this problem programmatically.

4: `icdiff`: <https://www.jefftk.com/icdiff>

## 6 | Conclusion

This thesis project implements a two-way converter, `chatconllu`, as a command-line tool written in Python for CHILDES CHAT and UD CoNLL-U format. Conversion from CHAT to CoNLL-U cleans the utterances with CHAT transcription codes, extracts information from the dependent tiers, associates them with their corresponding words in the cleaned sentences, and writes the sentences in CoNLL-U format. Back-conversion from CoNLL-U files outputs the original CHAT files, but the tabs for line continuations are missed during this process. Additionally, `chatconllu` can be used as an intermediate step to augment information to either annotation format. For instance, one can start with converting CHAT files to CoNLL-U format and pass them to any type of dependency parsers that work with this format. After that, the files can either be exported as a preliminary spoken treebank for human inspection, or converted back to CHAT format with added dependent tiers when asked to do so.

Moreover, as an attempt, a deterministic mapping to transform CHILDES morphosyntactic codes to UD values is created. Currently, this mapping oversimplifies the problem of tagset conversion by ignoring many non-deterministic cases and assigning uniform values, even though they can be decided using rule-based procedures.

In the future, `chatconllu` can be extended to support visualisations of dependency trees, which will ease the process of designing rules for tagset conversions by helping users experienced in dependency annotations to identify mal-formed and atypical dependency trees. Furthermore, in the hope of aiding independent researchers to tailor the program according to their needs, `chatconllu` can also be improved by giving the user more control over the output content.

# A | Morphosyntactic Mappings

## A.1 Part-of-Speech Mapping

**Table A.1:** Complete Set of Universal POS Tags

	Universal POS Tag	Description
<b>Open class words</b>	ADJ	adjective
	ADV	adverb
	INTJ	interjection
	NOUN	noun
	PROPN	proper noun
	VERB	verb
<b>Closed class words</b>	ADP	adposition
	AUX	auxiliary verb
	CCONJ	coordinating conjunction
	DET	determiner
	NUM	numeral
	PART	particle
	PRON	pronoun
	SCONJ	subordinating conjunction
<b>Other</b>	PUNCT	punctuation
	SYM	symbol
	X	other

All UPOS tags except SYM are used in chatconllu's POS mappings.

**Table A.2:** chatconllu Part-of-Speech Mapping 1—open-class words

Universal POS tag	MOR POS tag	Description
ADJ	adj	adjective
	adj:pred	adjective-predicative
ADV	adv	adverb
	adv:tem	adverb-temporal
INTJ	bab	babbling
	co	communicator
	fil	filler
	sing	singing
	wplay	wordplay
NOUN	n	noun
	n:let	noun-letter
	n:pt	noun-plurale tantum
	on	onomatopoeia
	onoma	onomatopoeia
PROPN	n:prop	proper noun
VERB	v	verb
	vimp	verb imperative

All MOR part-of-speech categories handled by the current version of chatconllu (part 1).

**Table A.3:** chatconllu Part-of-Speech Mapping 2—closed-class words

Universal POS tag	MOR POS tag	Description
ADP	post	postmodifier
	prep	preposition
	vpfx	preverb/verbal particles
AUX	aux	auxiliary
	cop	verb-copula
	mod	verb-modal
	v:aux	verb-auxiliary
CCONJ	coord	coordinator
DET	art	article
	det:art	determiner-article
	det:dem	determiner-demonstrative
	det:int	determiner-interrogative
	det:num	determiner-numeral
	det:poss	determiner-possessive
	prepart	preposition with article
	qn	quantifier
	quant	quantifier
NUM	num	numeral
PART	inf	infinitive
	neg	negative
	part	particle
PRON	pro:dem	pronoun-demonstrative
	pro:exist	pronoun-existential
	pro:indef	pronoun-indefinitive
	pro:int	pronoun-interrogative
	pro:obj	pronoun-object
	pro:per	pronoun-personal
	pro:poss	pronoun-possessive
	pro:refl	pronoun-reflexive
	pro:rel	pronoun-relative
	pro:sub	pronoun-subject
SCONJ	comp	complementizer
	conj	conjunction

All MOR part-of-speech categories handled by the current version of chatconllu (part 2).

**Table A.4:** chatconllu Part-of-Speech Mapping 3—other

Universal POS tag	MOR POS tag	Description
PUNCT	beg	MOR punctuation mark ‡
	cm	MOR punctuation mark ,
	end	MOR punctuation mark „
	bq	MOR punctuation mark “
	eq	MOR punctuation mark ”
	bq2	MOR punctuation mark ‘
	eq2	MOR punctuation mark ’
X	chi	child-invented form
	dia	dialect
	fam	family-specific form
	meta	metalinguistic use
	neo	neologism
	none	none
	phon	phonologically consistent form
	test	test words like ”wug”
	uni	Unibet transcription
	L2	second-language form

All MOR part-of-speech categories handled by the current version of chatconllu (part 3).



## A.2 Morphological Features Mapping

**Table A.5:** All MOR grammatical morpheme codes handled by the current version of chatconllu

Morpheme Code	UD feats
cp	Degree=Cmp
sp	Degree=Sup
m	Gender=Masc
f	Gender=Fem
cond	Mood=Cnd
imp	Mood=Imp
sub	Mood=Sub
subj	Mood=Sub
pl	Number=Plur
1p	Number=Plur Person=1
2p	Number=Plur Person=2
3p	Number=Plur Person=3
3sp	Number=Plur Person=3
sg	Number=Sing
1s	Number=Sing Person=1
2s	Number=Sing Person=2
12s	Number=Sing Person=2
3s	Number=Sing Person=3
13s	Number=Sing Person=3
23s	Number=Sing Person=3
poss	Poss=Yes
impf	Tense=Imp
fut	Tense=Fut
past	Tense=Past
pret	Tense=Past
pastp	Tense=Past VerbForm=Part
pp	Tense=Past VerbForm=Part
pres	Tense=Pres
ppre	Tense=Pres VerbForm=Part
presp	Tense=Pres VerbForm=Part
inf	VerbForm=Inf
pass	Voice=Pass

### A.3 Dependency Relation Mapping

**Table A.6:** All GR codes handled by the current version of chatconllu

GRA grammatical relation label	UD deprel
SUBJ	nsubj
CSUBJ	csbj
OBJ	obj
OBJ2	iobj
COMP	ccomp
PRED	xcomp
CPOBJ	acl:relcl
COBJ	ccomp
POBJ	obl
SRL	xcomp
JCT	advmod
CJCT	advcl
XJCT	advcl
NJCT	nmod
MOD	nmod
POSTMOD	amod
POSS	case
APP	appos
CMOD	ccomp
XMOD	acl
DET	det
QUANT	det
PQ	det
AUX	aux
NEG	advmod
INF	mark
LINK	mark
TAG	parataxis
COM	discourse
BEG	discourse, vocative or parataxis conditionally
END	parataxis
INCROOT	root
OM	discourse:omission
PUNCT	punct
LP	punct
BEGP	punct
ENDP	punct
ROOT	root
NAME	flat:name
DATE	flat
ENUM	conj
CONJ	conj
COORD	cc

# References

- Ameka, F. (1992). Interjections: The universal yet neglected part of speech. *Journal of Pragmatics*, 18(2), 101–118. [https://doi.org/https://doi.org/10.1016/0378-2166\(92\)90048-G](https://doi.org/https://doi.org/10.1016/0378-2166(92)90048-G) (page 18)
- Bouma, G., van Noord, G., & Malouf, R. (2000). Alpino: Wide-coverage Computational Analysis of Dutch. (Pages 2, 12).
- Brown, R. (1973). Development of the first language in the human species. *American Psychologist*, 28(2), 97–106. <https://doi.org/10.1037/h0034209> (pages 2, 4, 10)
- Buchholz, S., & Marsi, E. (2006). CoNLL-X Shared Task on Multilingual Dependency Parsing, In *Proceedings of the tenth conference on computational natural language learning (CoNLL-x)*, New York City, Association for Computational Linguistics. <https://aclanthology.org/W06-2920>. (Page 5)
- Liu, Z., & Prud'hommeaux, E. (2021). Dependency Parsing Evaluation for Low-resource Spontaneous Speech, In *Proceedings of the second workshop on domain adaptation for nlp*. (Pages 2, 12, 22).
- Macwhinney, B. (2000). The CHILDES Project: Tools for Analyzing Talk (third edition): Volume I: Transcription format and programs, Volume II: The database. *Computational Linguistics - COLI*, 26, 657–657. <https://doi.org/10.1162/coli.2000.26.4.657> (pages 1, 3, 4, 10, 13, 16, 18, 21)
- Matías Grioni. (2018). *pyconll Documentation*. <https://pyconll.github.io/>. (Pages 9, 14)
- Nivre, J. (2015). Towards a Universal Grammar for Natural Language Processing (A. Gelbukh, Ed.). In A. Gelbukh (Ed.), *Computational linguistics and intelligent text processing*, Cham, Springer International Publishing. (Page 1).
- Nivre, J., de Marneffe, M.-C., Ginter, F., Goldberg, Y., Hajič, J., Manning, C. D., McDonald, R., Petrov, S., Pyysalo, S., Silveira, N., Tsarfaty, R., & Zeman, D. (2016). Universal Dependencies v1: A Multilingual Treebank Collection, In *Proceedings of the tenth international conference on language resources and evaluation (LREC'16)*, Portorož, Slovenia, European Language Resources Association (ELRA). <https://aclanthology.org/L16-1262>. (Page 1)
- Nivre, J., de Marneffe, M.-C., Ginter, F., Hajič, J., Manning, C. D., Pyysalo, S., Schuster, S., Tyers, F., & Zeman, D. (2020). Universal Dependencies v2: An Evergrowing Multilingual Treebank Collection, In *Proceedings of the 12th language resources and evaluation conference*, Marseille, France, European Language Resources Association. <https://aclanthology.org/2020.lrec-1.497>. (Page 1)
- Odijk, J., Dimitriadis, A., van der Klis, M., van Koppen, M., Otten, M., & van der Veen, R. (2018). The AnnCor CHILDES Treebank, In *Proceedings of the eleventh international conference on language resources and evaluation (lrec 2018)*. European Language Resources Association (ELRA). (Pages 2, 12, 22).
- Parisse, C., & Normand, M.-T. (2000). Automatic disambiguation of morphosyntax in spoken language corpora. *Behavior research methods, instruments, computers : a journal of the Psychonomic Society, Inc*, 32, 468–81 (page 1).
- Petrov, S., Das, D., & McDonald, R. (2012). A Universal Part-of-Speech Tagset, In *Proceedings of the eighth international conference on language resources and evaluation (LREC'12)*, Istanbul, Turkey, European Language Resources Association (ELRA). [http://www.lrec-conf.org/proceedings/lrec2012/pdf/274\\_Paper.pdf](http://www.lrec-conf.org/proceedings/lrec2012/pdf/274_Paper.pdf). (Page 16)
- Sagae, K., MacWhinney, B., & Lavie, A. (2004a). Adding Syntactic Annotations to Transcripts of Parent-Child Dialogs, In *Proceedings of the fourth international conference on language resources and evaluation (LREC'04)*, Lisbon, Portugal, European Language Resources Association (ELRA). <http://www.lrec-conf.org/proceedings/lrec2004/pdf/749.pdf>. (Page 19)
- Sagae, K., Macwhinney, B., & Lavie, A. (2004b). Automatic Parsing of Parent-Child Interactions. <https://doi.org/10.1184/R1/6613649.v1> (page 1)
- Sagae, K., Davis, E., Lavie, A., Macwhinney, B., & Wintner, S. (2007). High-accuracy annotation and parsing of CHILDES transcripts. *Proceedings of the Workshop on*

*Cognitive Aspects of Computational Language Acquisition*.  
<https://doi.org/10.3115/1642025.1642029> (page 19)

Straka, M., Hajič, J., & Straková, J. (2016). UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing, In *Proceedings of the tenth international conference on language resources and evaluation (LREC'16)*, Portorož, Slovenia, European Language Resources Association (ELRA). <https://aclanthology.org/L16-1680>. (Pages 1, 15)

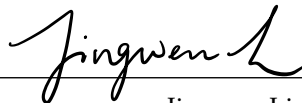
Zeman, D., Nivre, J., Abrams, M., Ackermann, E., Aepli, N., Aghaei, H., Agić, Ž., Ahmadi, A., Ahrenberg, L., Ajede, C. K., Aleksandravičiūtė, G., Alfina, I., Antonsen, L., Aplonova, K., Aquino, A., Aragon, C., Aranzabe, M. J., Arnardóttir, H., Arutie, G., ... Zhuravleva, A. (2020). Universal Dependencies 2.7 [LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University]. <http://hdl.handle.net/11234/1-3424>. (Page 5)

# Declaration of Authorship

Hiermit versichere ich, dass ich die vorstehende Bachelorarbeit selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe und dass die Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen ist und dass die Arbeit weder vollständig noch in wesentlichen Teilen bereits veröffentlicht wurde sowie dass das in Dateiform eingereichte Exemplar mit den eingereichten gebundenen Exemplaren übereinstimmt.

Tübingen, den

05.10.2021



Jingwen Li