Yauti: A Tool for Morphosyntactic Analysis of Nheengatu within the Universal Dependencies Framework

Leonel Figueiredo de Alencar[1]

[1]Universidade Federal do Ceará (UFC), Brazil

Av. da Universidade 2683 – 60.020-181 – Fortaleza – CE – Brazil

leonel.de.alencar@ufc.br

Abstract. *This paper reports on Yauti, a rule-based morphosyntactic analyzer for the endangered Brazilian indigenous language Nheengatu. Its goal is to generate analyses in the UD framework's CoNLL-U format. It has been developed on par with the construction of the Nheengatu treebank of the UD collection. In sentences only consisting of known and unambiguous words, the tool generally delivers good results. It obtained a LAS score of 73.2% in a version of the Nheengatu UD treebank with all 1022 sentences automatically provided with XPOS tags and a special annotation to handle non-lexicalized words.*

1. Introduction

Natural language processing has attained near or state-of-the-art results in Brazil in the last two decades. The focus has been on Portuguese, placing it among the languages with the highest Digital Language Support (DLS) Level [Simons et al. 2022, Eberhard et al. 2023]. Despite that, the enormous linguistic diversity in the country, represented by about 150 indigenous languages still alive, has been practically ignored. Only recently has attention been paid to these languages, with the creation of resources such as treebanks and the implementation of computational analysis tools, e.g., [Galves et al. 2017, da Silva Facundes et al. 2021, Gerardi et al. 2021, Martín Rodríguez et al. 2022]. Worldwide, the computational processing of minority languages has enjoyed a growing interest, also on the part of information technology giants, e.g., [Bapna et al. 2022].

This paper describes a contribution to bridging the digital divide that endangers the survival of minority languages in Brazil. We present Yauti, a tool for morphosyntactic analysis of Nheengatu (Modern Tupí) within the Universal Dependencies framework (henceforth UD) [Nivre et al. 2016, de Marneffe et al. 2021].[1] Nheengatu was the most spoken language in the Brazilian Amazon until the middle of the 19th century [Navarro et al. 2017]. Today it is threatened with extinction [Eberhard et al. 2023]. It is the indigenous language that has been most widely learned across Brazil by language enthusiasts or in revitalization initiatives as a means of affirming ethnic identity. It also stands out for its well-documented history spanning four centuries [Moore et al. 1994, Rodrigues 1996, Freire 2011, Rodrigues and Cabral 2011, Moore 2014].

Notwithstanding its cultural, social, and historical significance, there was no publicly available corpus or computational tool for processing Nheengatu until recently. Accordingly, Nheengatu ranks with a score of 0.07 at the bottom of the DLS

[1]Yauti is available at `https://github.com/CompLin/nheengatu`.

scale, in a seemingly infinite distance from Portuguese with 0.96 and English with 1.00. Recent efforts to revert this situation include [de Alencar 2021], who reports on a machine translation prototype and a small semantic treebank for stative sentences. [Alexandre et al. 2021a, Alexandre et al. 2021b] deal with a simplistic part-of-speech tagger and a tagged corpus without any syntactic disambiguation.

Yauti represents a much more ambitious initiative. POS taggers, parsers, and annotated corpora are of vital importance not only for developing language technology applications, but also for language documentation, language learning and instruction, and typological investigation. A by-product of the development of Yauti is the UD_Nheengatu-CompLin treebank (henceforth UDTB), the second largest of a Brazilian indigenous language in the UD collection. The tool enabled the treebank to grow relatively fast from 196 sentences in Version 2.11 of 11/15/22 to 860 in Version 2.12 of 5/15/23. As of writing this paper, the development version of UDTB features 1022 sentences.

Section 2 deals with the annotation task according to the UD scheme. Section 3 describes Yauti's different components and explains how it performs sentence annotations. After reporting on the evaluation results in Section 4, we point out in Section 5 directions for further research.

2. The Annotation Task

A UD treebank's annotation scheme can be divided into two components. While the first comprises principles considered universal and that every language in the collection must adhere to, the second consists of specific requirements for a particular treebank of a particular language. Both incorporate specifications related to the different levels of morphosyntactic analysis, from orthographic word and syntactic word definitions to parts of speech, feature structures, and syntactic relations. In addition, the treebank file(s) must conform to the CoNLL-U format. A validation program verifies compliance with all of these requirements and grants valid status to a treebank only if it does not violate any of these specifications.[2]

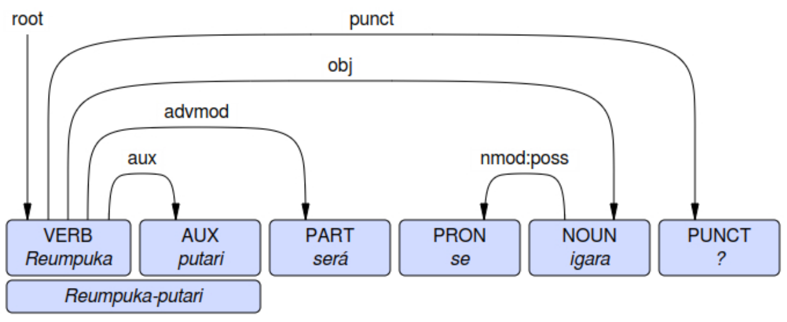
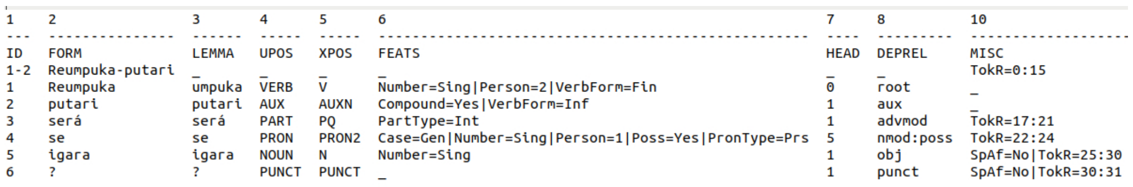


Figure 1. Dependency graph of example (3)

Some of these well-formed conditions are common to different syntactic frameworks, such as the limitation of a single subject per clause. Other requirements seem theory-specific, for example, the prohibition that nouns or pronouns function as adverbial modifiers. UD establishes limited inventories of parts of speech (UPOS) and syntactic relations (DEPREL), which a given language need not exhaust.

[2]https://github.com/UniversalDependencies/tools

Generating an analysis for a Nheengatu sentence conforming to the annotation scheme of UDTB requires filling in nine of the ten fields specified in the CoNLL-U format[3] with lexical-grammatical information for each syntactic word; see Figures 1 and 2.[4] UDTB does not specify yet enhanced dependency relations [Schuster and Manning 2016], housed in column #9.



```
1    2                3       4      5      6                                                    7     8          10
---  ---------------  ------  -----  -----  ---------------------------------------------------  ----  ---------  ------------------
ID   FORM             LEMMA   UPOS   XPOS   FEATS                                                HEAD  DEPREL     MISC
1-2  Reumpuka-putari  _       _      _      _                                                    _     _          TokR=0:15
1    Reumpuka         umpuka  VERB   V      Number=Sing|Person=2|VerbForm=Fin                    0     root       _
2    putari           putari  AUX    AUXN   Compound=Yes|VerbForm=Inf                            1     aux        _
3    será             será    PART   PQ     PartType=Int                                         1     advmod     TokR=17:21
4    se               se      PRON   PRON2  Case=Gen|Number=Sing|Person=1|Poss=Yes|PronType=Prs  5     nmod:poss  TokR=22:24
5    igara            igara   NOUN   N      Number=Sing                                          1     obj        SpAf=No|TokR=25:30
6    ?                ?       PUNCT  PUNCT  _                                                    1     punct      SpAf=No|TokR=30:31
```

Figure 2. Example of annotation in the CoNLL-U format automatically generated by Yauti for example (3).

Table 1[5] provides basic statistics of UDTB and MYTH (see Section 4) provided by the `conllu-stats.pl` script from the UD project tools repository. They allow one to dimension the complexity of the task of annotating a sentence in Nheengatu according to the annotation scheme of UDTB.

Table 1. Statistics of UDTB and MYTH

Dataset	Sents	Tokens	Words	Fused	Lemmas	Forms	Fusions	Tags	Feats	Deps
UDTB	1022	10082	10181	99	1075	1450	72	15	66	36
MYTH	43	580	583	3	154	175	3	13	46	24

The first annotation subtask is to determine the syntactic words. Within the context of UD, tokenization requires not just splitting sentences on orthographic words and punctuation. In the cases computed on the Fused column of Table 1, a single token maps onto two different syntactic words. In (1)-(3) from UDTB, the fused words are in boldface.

(1) *Amaã**ntu***. 'I'm just looking.' (Avila2021:0:0:195)

(2) *Setá reté pirá parana**me***. 'There are many fish in the river.' (Avila2021:0:0:341)

(3) *Reumpuka-**putari** será se igara?* 'Do you want to break my canoe?' (Avila2021:0:0:498)

In (1) and (2), the adverb and postposition cliticize to their heads. In (3), the auxiliary incorporates into the main verb. Yauti handles both cases through a two-step process. First, it splits tokens on white space and separates punctuation. Second, it traverses the

[3]`https://universaldependencies.org/format.html`

[4]Figure 1 was generated from the CoNLL-U annotation in UDTB by the viewer at `https://urd2.let.rug.nl/~kleiweg/conllu/`. For presentation purposes, we manually edited the program output displayed in Figure 2. *TokR* and *SpAf* are abbreviations for *TokenRange* and *SpaceAfter*, respectively.

[5]*Sents* and *Deps* refer to the number of sentences and syntactic relations. The values from the Lemmas column onwards exclude repetitions.

resulting list of strings and splits off each of a predefined list of suffixes, signaling it with a specific mark to indicate whether it is a compound member or a clitic.

Yauti, using the CoNLL-U Parser[6] Python library, generates an object of the `Token` class for each orthographic word and each syntactic word. The identity of each `Token` object is an integer or a range of integers, as in Figure 2. Each syntactic word must receive a label from the UPOS and XPOS inventories. The feature structures (FEATS) of each syntactic word must be assigned. The head of the word and the type of syntactic relation (DEPREL) it maintains with that head must be determined.

This last subtask gets especially tricky in complex sentences with coordinated and/or subordinated clauses, clausal complements, etc. Commas, dashes, semicolons, colons, and quotation marks attach to the highest node of these constituents. Failure to identify this highest node triggers the wrong annotation of its dependent punctuation marks. Currently, UDTB has 1913 punctuation marks, totaling 13 different types.

The last column of the CoNLL-U format specifies, via the *TokenRange* attribute, the span of the syntactic word and, through the *SpaceAfter* attribute, whether it is followed by white space or not. As the first situation is the most general, this attribute is only used when the value is negative. Its absence implies a positive value. Other attributes housed in this last column are *Orig* and *OrigLang*, which specify, respectively, the original form and the language of origin of loanwords not yet lexicalized in Nheengatu.

3. Program Construction and Operation

It is evident from the examples presented above that annotating sentences according to the UDTB scheme is a complex task. Therefore, we decompose it into a series of subtasks, performed by different functions implemented in a series of Python modules.

Yauti's starting point is a glossary in JSON format with 1552 entries, generated from a raw text file that a linguist without programming skills can easily edit; see example entry in (4). The first version of this glossary only contained about 850 entries. We created it by extracting the raw text from [de Almeida Navarro 2016]'s glossary, after some post-editing to remove page numbers, correct inconsistent formatting, etc. The circa 700 new entries are a manual sample from the over 8000 entries in [Avila 2021].

(4) ***maã*** 5 (*verb*) - to see, to look

Navarro's word classification underwent profound changes to handle annotation in the UD scheme. Instead of his original system of 12 classes and 6 pronoun subclasses, Yauti, based, e.g., on [da Cruz 2011] and [Avila 2021], adopts a much more granular inventory with 83 XPOS tags. For example, instead of a single tag for all particles, Yauti has 20 tags for different particle types. Particles play a fundamental role in Nheengatu, whose verb inflection is relatively poor. Pronouns, determiners, and adverbs also have very high granular labels. Pronouns and determiners are divided into 15 subcategories, e.g., INDQ for indefinite quantitative pronouns. Adverbs are classified into 22 subcategories, such as ADVT, ADVC, and ADVS, for temporal, locative, and intensity adverbs.

The glossary in JSON format feeds a rule-based morphological generator, con-

[6] `https://pypi.org/project/conllu/`

sisting of functions that take as parameters lemma, part of speech, and inflectional class information, deriving conjugated verbs, pluralized nouns and pronouns, and words with relational prefixes.[7] This generator populates a full-form lexicon of Nheengatu in the form of a Python dictionary mapping word forms to a list of `[lemma, tags]` pairs, where `tags` is a sequence of tags separated by the plus sign. For example, the word forms *amaã* 'I see' and *reumpuka* 'you break' from (1) and (3) map to `[['maã', 'V+1+SG']]` and `[['umpuka', 'V+2+SG']]`, respectively. The ambiguous word form *setá*, which functions as a third class verb (V3) in (2), maps to `[['setá', 'ADVS'], ['setá', 'INDQ'], ['setá', 'V3+NCONT']]`.

Using the information encoded in the lexicon, Yauti performs an initial filling of different annotation fields of the `Token` objects, notably UPOS and FEATS. It assigns the feature `Clitic=Yes` or `Compound=Yes` to suffixes that constitute syntactic words, as in (1)-(3). For some word classes, such as ADP, INTJ, CCONJ, and SCONJ, the syntactic relation is already specified at this stage. Additionally, Yauti parses productive morphological derivations. One noteworthy aspect of Nheengatu is the application of aspectual suffixes not only to auxiliary and main verbs, but also to nouns, adverbs, etc.

Yauti then inserts each `Token` object in a `TokenList`. Then it traverses this list token by token, executing functions that fill in the head and syntactic relations, based on available information from the current token and preceding and succeeding tokens.

To this end, Yauti makes use of general syntactic patterns of Nheengatu. Let's look at some examples. A noun that does not govern an adposition and precedes a verb generally functions as its subject, while an analogous noun following a verb functions as its object. The closest noun to the left of a postposition constitutes its head. This noun, in turn, is linked to the nearest verb through the oblique syntactic relation, noun-dependent adpositional phrases being rare. In noun sequences, the one farthest to the left is a possessive modifier of its next neighbor to the right, which, in turn, is a possessive modifier of the next noun in the sequence, and so on.

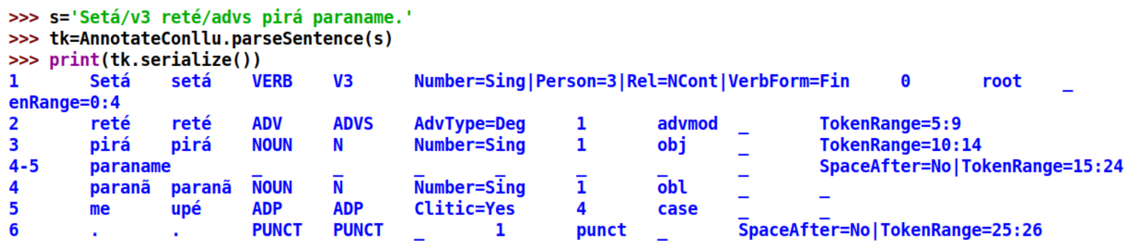
Subordinating conjunctions (SCONJ) behave analogously to postpositions: the closest verb to the left is its head, and this verb is linked as an adverbial clause modifier to another verb. Yauti adopts the following heuristics to identify the head of a verb governing a SCONJ: try to find a verb in a preceding matrix clause, if there is none, try to find one in a subsequent clause. Under the scope of the negation particle, however, subordinating conjunctions and the relative pronoun precede their governing verb.

Very typical of Nheengatu are clauses in parataxis, where the first verb governs the second, which governs the next one, and so on. Yauti often manages to handle this construction. However, it fails to correctly analyze clausal complements (*ccomp* and *xcomp* in UD) since Nheengatu has no complementizers or infinitives, and information on verb valence is presently unavailable. Another still unsolved challenge is correctly identifying the head of a noun between two verbs linked via parataxis. It can either be the *obj* of the first verb or the *nsubj* of the second. Yauti always chooses the latter option, which is sometimes incorrect.

As there are no comprehensive formalized descriptions of Nheengatu, the con-

[7]Relational prefixes encode, e.g., syntactic contiguity (CONT) or non-contiguity (NCONT) between a head and its dependent [Rodrigues and Cabral 2011].

struction of the syntactic annotation algorithm proceeded incrementally. We initially made a very simple version of the algorithm to handle basic cases like the SVO pattern in simple sentences. We then applied it to increasingly complex and diverse examples, observing the mistakes made and correcting the code to avoid them. The current algorithm is the result of this process after the annotation of more than 1000 sentences (Table 1). The vocative is an example of a syntactic relation whose annotation has only recently become automated. Before that, Yauti incorrectly assigned the *nsubj* or *obj* relation to verb dependents in the vocative function, following the most general SVO pattern. Now a rule corrects this initial annotation, provided some conditions are met.



```
>>> s='Setá/v3 reté/advs pirá paraname.'
>>> tk=AnnotateConllu.parseSentence(s)
>>> print(tk.serialize())
1       Setá    setá    VERB    V3      Number=Sing|Person=3|Rel=NCont|VerbForm=Fin     0       root    _
enRange=0:4
2       reté    reté    ADV     ADVS    AdvType=Deg     1       advmod  _       TokenRange=5:9
3       pirá    pirá    NOUN    N       Number=Sing     1       obj     _       TokenRange=10:14
4-5     paraname        _       _       _       _       _       _       SpaceAfter=No|TokenRange=15:24
4       paranã  paranã  NOUN    N       Number=Sing     1       obl     _       _
5       me      upé     ADP     ADP     Clitic=Yes      4       case    _       _
6       .       .       PUNCT   PUNCT   _       1       punct   _       SpaceAfter=No|TokenRange=25:26
```

Figure 3. Parsing example (2) with Yauti in the Python IDLE shell

Yauti's central function is `parseSentence`, which takes as an argument a string with or without part-of-speech tags and other abbreviations, as explained below. This function returns an object of the `TokenList` class. Thus, the program can be used in batch processing as a component of a pipeline, generating analyses in the CoNLL-U format, following the UDTB annotation scheme for the sentences given as input.

However, because of ambiguity, the results generated by the fully automatic mode of the program are in general precarious. For each syntactic word with *n* XPOS tags, the program generates *n* `Token` objects. Thereafter, `TokenList` creation rules operate on these objects, generating chain errors. To address this problem, an automatic part-of-speech tag disambiguator is under development. For the time being, it is in interactive mode that the program is most effective. In a typical interaction, the user types in an example in the Python shell, executing the `parseSentence` function. Yauti generates the respective analysis. The user checks which words are ambiguous and manually eliminates the ambiguities by specifying the corresponding tag, as in Figure 3. Next, the user reruns `parseSentence` on the tagged sentence, manually correcting detected errors, e.g., after inspecting the visualization of the sentence or executing the validation program. In Figure 3, Yauti incorrectly assigned node 3 the *obj* instead of the *nsubj* relation.

In addition to disambiguating tags, `parseSentence` accepts two other types of abbreviations. First, `@` assigns a word the *root* syntactic relation. This generally improves the annotation of verbless sentences, which are common in Nheengatu. They can be tricky for Yauti to deal with, as any content word can be the main predicate of the sentence and act as *root*. Second, we have functions for parsing words the morphological analyzer does not know about and are inappropriate for Yauti's lexical database. These unknown words fall into two groups: (i) Portuguese words in code-switching, proper nouns, and non-lexicalized interjections, e.g., onomatopoeias; (ii) productive morphological derivations, e.g., collective nouns, words with degree, privative or aspectual suffixes, etc.

4. Evaluation of Yauti

UDTB is the only Nheengatu syntactic treebank available. To evaluate Yauti, therefore, we could not count on a previous gold standard. Instead, to assess how useful the tool can be for annotating texts in Nheengatu, we resorted first to UDTB (Table 1). Given the grammatical and lexical diversity of this treebank, compiled from excerpts of different genres from different regions and diachronic stages, we expect that the average performance in this dataset should reflect in the annotation of other sentences from the same sources or other texts of more or less similar characteristics.

Parsing Nheengatu is especially challenging due to the lack of spelling standardization and limited computational resources. On the other hand, the current lexicon of the morphological analyzer only covers a fraction of [Avila 2021]. Another still unresolved issue is the ambiguity resolution. Thus, we have restricted the evaluation task to three main dimensions by isolating the effects of the unknown word analysis and ambiguity resolution subtasks. The first dimension is the performance in the standard LAS and UAS metrics [Straka et al. 2016, Nivre and Fang 2017, Straka 2018]. The second is the splitting of tokens into syntactic words. The third is morphological analysis, encompassing both lemmatization and UPOS and XPOS assignment. Additionally, we computed accuracy for the *SpaceAfter* attribute.

To perform this evaluation, we implemented a Python code that traverses each `TokenList` of UDTB and extracts, from each `Token` object, a triple in the format `(form, tag, spaceafter)`. The second member of this triple can either be the lower-cased tag from the XPOS field or an abbreviation of the name of a `Token` object construction function, prefixed with = and possibly followed by a series of colon-separated strings in the form of `k|v`, where `k` is a keyword argument and `v` its value. From these triples, another function constructs a valid input for the `parseSentence` function. (5)-(7) are test sentences automatically constructed from the information encoded in UDTB. In (5), `=v` and `=n` trigger the creation of verb and noun `Token` objects for Portuguese loanwords not registered in [Avila 2021]. The function `=hab` in (6) and (7) handles the frequentative aspect suffix *-wara*, which attaches to the verb *sú* 'to go' in (11) and to the adverb *iké* 'here' in (7). The named parameters `x` and `a` handle the XPOS and the accentuation of the final vowel of the base form.

(5) *Presizu/****nec*** *aintá/****pron*** *uistudari/****=v*** *português/****=n*** *upé/****adp***. 'It's necessary that they study in Portuguese.' (MooreFP1994:0:0:8)

(6) *Asuwara/****=hab:x|v:a|t*** *mikití/****advc***. 'I always go there.' (Avila2021:0:0:432)

(7) *Ikewara/****=hab:x|advdx:a|t*** *aikú/****cop***, *reyuri/****v*** *ramé/****sconj***, *resika/****v*** *se/****pron2*** *piri/****adp***. 'I'm always here, when you come, come visit me.' (Avila2021:0:0:429)

Applied to all 1022 thusly annotated sentences from UDTB, Yauti obtained 80.0 and 73.2 in the UAS and LAS metrics, respectively (Table 2). These values are similar to those achieved in some treebanks by the deep neural parser of the UDPipe 2.0 prototype in the CoNLL 2018 UD Shared Task competition [Straka 2018]. For example, this parser, representative of the state-of-the-art in dependency parsing, achieved a UAS of 78.66 and LAS of 74.25 in the Galician-TreeGal treebank. In the other two principal dimensions, Yauti delivered a much higher performance. Specifically, it achieved an accuracy score of 99.7% and 98.4% in the assignment of lemmas and features, respectively, attaining 100% accuracy in tokenization. It is worth noting that several lemma and feature errors have turned out to be incorrect annotations of UDTB. Yauti also exhibited high accuracy in the *SpaceAfter* attribute, with a score of 99.5%.

Table 2. Performance of Yauti in two data sets

Dataset	LAS	UAS	Lemmas	Feats
UDTB	73.2	80.0	99.7	98.5
MYTH	71.0	76.3	96.6	96.8

This experiment also revealed 96 XPOS and 204 UPOS discrepancies compared to UDTB, many of which were actually incorrect annotations in the treebank. Most errors Yauti commits in this domain involve distinguishing pronouns from determiners, on the one hand, and auxiliary from main verbs, on the other. In the lexicon, these distinctions are underspecified. Yauti tries to guess the correct tags during the construction of the `TokenList` object. However, the accuracy levels show that the algorithm needs improvement in this aspect.

As a first step to test performance on unseen data, we applied the tool to the myth "How the Night Appeared" [de Magalhães 1876, pp. 163-171]. We modernized the spelling but kept the original punctuation intact.[8] This story contains relatively many obsolete forms. We manually updated the JSON glossary with 34 lemmas hitherto unknown to Yauti. We also manually annotated the sentences, as in (5)-(7). We corrected Yauti's output and compared this gold standard to the test version. Table 2 displays the main results.

5. Final Remarks

We have encountered several annotation errors we could not yet prevent due to time constraints or a lack of fuller understanding of the respective phenomena. One of the primary issues we face is with verb-subject clauses (Figure 3). Cross-linguistically, this construction relates to unaccusativity, which is difficult to detect automatically. Another challenge comes from verbless sentences, where any content word can act as *root*. A further difficulty is the indirect object (*iobj*), defined in UD theory as a verb's core argument in addition to an *obj* or a *ccomp*. While identifying it may be challenging in the general case, it seems clear-cut in languages such as English that feature double objects or in languages like German that use a dative case. In UDTB, a noun governing a dative postposition is an *iobj* whenever the same verb governs an *obj* or a *ccomp*. However, as we saw, the latter relation is hard to identify in Nheengatu. We intend to address the issues identified so far soon in the hope of improving Yauti's performance.

The evaluation experiment showed that Yauti is useful not only to annotate new sentences but also to check existing annotations for consistency. As UDTB constantly increases, we hope to train a neural parser on the data with UDPipe 2.0 and compare its performance with the improved rule-based approach.

[8]We owe the orthography adaptation to the philological expertise of Marcel Twardowsky Avila (p.c.).

References

Alexandre, D. M., Gurgel, J. L., and de A. Araripe, L. F. (2021a). Compilação de um corpus etiquetado da Língua Geral Amazônica. In *Anais do XIII Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana*, pages 427–431, Porto Alegre, RS, Brasil. SBC.

Alexandre, D. M., Gurgel, J. L., and de Alencar Araripe, L. F. (2021b). Nheentiquetador: Um etiquetador morfossintático para o sintagma nominal do nheengatu. *Revista Encontros Universitários da UFC*, 6:1–13.

Avila, M. T. (2021). *Proposta de dicionário nheengatu-português*. PhD thesis, Faculdade de Filosofia, Letras e Ciências Humanas da Universidade de São Paulo.

Bapna, A., Caswell, I., Kreutzer, J., Firat, O., van Esch, D., Siddhant, A., Niu, M., Baljekar, P., Garcia, X., Macherey, W., Breiner, T., Axelrod, V., Riesa, J., Cao, Y., Chen, M. X., Macherey, K., Krikun, M., Wang, P., Gutkin, A., Shah, A., Huang, Y., Chen, Z., Wu, Y., and Hughes, M. (2022). Building machine translation systems for the next thousand languages. Technical report, Google Research.

da Cruz, A. (2011). *Fonologia e gramática do nheengatú: A língua falada pelos povos Baré, Warekena e Baniwa*. LOT, Utrecht.

da Silva Facundes, S., de Freitas, M. F. P., and de Lima-Padovani, B. F. S. (2021). Number expression in Apurinã (Arawák). In Hämäläinen, M., Partanen, N., and Alnajjar, K., editors, *Multilingual Facilitation*, pages 31–42. University of Helsinki Library, Helsinki.

de Alencar, L. F. (2021). Uma gramática computacional de um fragmento do nheengatu / A computational grammar for a fragment of nheengatu. *Revista de Estudos da Linguagem*, 29(3):1717–1777.

de Almeida Navarro, E. (2016). *Curso de Língua Geral (nheengatu ou tupi moderno): A língua das origens da civilização amazônica*. Centro Angel Rama da Faculdade de Filosofia, Letras e Ciências Humanas da Universidade de São Paulo, São Paulo, second edition.

de Magalhães, J. V. C. (1876). *O selvagem*. Typographia da Reforma, Rio de Janeiro.

de Marneffe, M.-C., Manning, C. D., Nivre, J., and Zeman, D. (2021). Universal Dependencies. *Computational Linguistics*, 47(2):255–308.

Eberhard, D. M., Simons, G. F., and Fennig, C. D., editors (2023). *Ethnologue: Languages of the World*. SIL International, Dallas, twenty-sixth edition.

Freire, J. R. B. (2011). *Rio Babel: A história das línguas na Amazônia*. EdUERJ, Rio de Janeiro, second edition.

Galves, C., Sandalo, F., Sena, T. A. d., and Veronesi, L. (2017). Annotating a polysynthetic language: From Portuguese to Kadiwéu. *Cadernos de Estudos Linguísticos*, 59(3):631–648.

Gerardi, F. F., Reichert, S., and Aragon, C. C. (2021). TuLeD (tupían lexical database): introducing a database of a South American language family. *Language Resources and Evaluation*, 55(4):997–1015.

Martín Rodríguez, L., Merzhevich, T., Silva, W., Tresoldi, T., Aragon, C., and Gerardi, F. F. (2022). Tupían language ressources: Data, tools, analyses. In *Proceedings of the 1st Annual Meeting of the ELRA/ISCA Special Interest Group on Under-Resourced Languages*, pages 48–58, Marseille, France. European Language Resources Association.

Moore, D. (2014). Historical development of Nheengatu (Língua Geral Amazônica). In Mufwene, S. S., editor, *Iberian Imperialism and Language Evolution in Latin America*, pages 108–142. University of Chicago Press, Chicago.

Moore, D., Facundes, S., and Pires, N. (1994). Nheengatu (Língua Geral Amazônica), its history, and the effects of language contact. In *Proceedings of the Meeting of the Society for the Study of the Indigenous languages of the Americas, July 2-4, 1993 and the Hokan-Penutian workshop, July 3, 1993*, Report / Survey of California and other Indian Languages ; 8, pages 93–118, Berkeley, CA. [University of California].

Navarro, E., Ávila, M., and Trevisan, R. (2017). O nheengatu, entre a vida e a morte: A tradução literária como possível instrumento de sua revitalização lexical. *Revista Letras Raras*, 6(2):9–29.

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., and Zeman, D. (2016). Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Nivre, J. and Fang, C.-T. (2017). Universal Dependency evaluation. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 86–95, Gothenburg, Sweden. Association for Computational Linguistics.

Rodrigues, A. D. (1996). As línguas gerais sul-americanas. *Papia*, 4(2):6–18.

Rodrigues, A. D. and Cabral, A. S. A. C. (2011). A contribution to the linguistic history of the língua geral amazônica. *ALFA: Revista de Linguística*, 55(2).

Schuster, S. and Manning, C. D. (2016). Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2371–2378, Portorož, Slovenia. European Language Resources Association (ELRA).

Simons, G. F., Thomas, A. L. L., and White, C. K. K. (2022). Assessing digital language support on a global scale. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4299–4305, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Straka, M. (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.

Straka, M., Hajič, J., and 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)*, pages 4290–4297, Portorož, Slovenia. European Language Resources Association (ELRA).