# 14 Exploring parallel corpora with STUnD: A Search Tool for Universal Dependencies

Arianna Masciolini
University of Gothenburg

Herbert Lange
University of Gothenburg

Márton András Tóth
University of Gothenburg

We introduce STUnD (Search Tool for Universal Dependencies), a corpus search tool designed to facilitate working with parallel data. STUnD employs a query language that allows describing syntactic structures and specifying divergence patterns, which in turn make it possible to look for systematic differences between texts. Furthermore, the tool can automatically detect the differences between two similar documents. To achieve all this, STUnD leverages Universal Dependencies (UD), a cross-lingually consistent standard for morphosyntactic annotation. Input can consist of pre-annotated UD treebanks or raw text, which the tool automatically processes through a third-party parser. As demonstrated in the case study included in the present chapter, STUnD is especially well-suited for comparing syntactic structures across languages, with applications in the context of typology and translation studies. Other use cases include retrieving grammatical errors from parallel learner corpora and comparing different analyses of the same text.

## 1 *Introduction*

This chapter introduces STUnD (Search Tool for Universal Dependencies), a web-based corpus search tool.[1] Unlike most other such services, STUnD is designed with parallel texts in mind.[2] Its interface makes it possible to inspect and compare two sentence-aligned corpora side by side. Upon running a query, these are processed in parallel, retrieving pairs of semantically equivalent sentence fragments matching the given query. STUnD's

1 A live demo of the tool is available at harisont.github.io/STUnD.
2 The acronym "STUnD" initally stood for *Sökverktyg för Tvåspråkiga Universal Dependencies trädbanker*, Swedish for 'Search Tool for Bilingual Universal Dependencies treebanks'.

HUM INFRA Arianna Masciolini, Herbert Lange & Márton András Tóth. 2025. Exploring parallel corpora with STUnD: A Search Tool for Universal Dependencies. In Gerlof Bouma, Dana Dannélls, Dimitrios Kokkinakis & Elena Volodina (eds.), *Huminfra handbook: Empowering digital and experimental humanities* (NEALT Proceedings Series 59), 455–503. University of Tartu Library. DOI: 10.58009/aere-perennius0183 

pattern matching language also supports so-called *divergence patterns*, which let users provide additional corpus-specific details to look for systematic differences between the two input texts. When STUnD is used on a bilingual Swedish-English corpus, for instance, a simple query allows retrieving all Swedish present-tense verb constructions and their English translations, whereas a divergence pattern enables looking for the specific instances where a Swedish *enkelt presens* 'simple present' is rendered as a *present continuous* (*be* + *-ing*) in the English translation.[3] To achieve this, STUnD relies on rich morphosyntactic annotation following the cross-lingually consistent Universal Dependencies (UD) standard (de Marneffe et al. 2021). Input corpora are either pre-annotated UD *treebanks*, i.e. collections of dependency-syntactic trees representing individual sentences, or raw text files that the tool automatically processes through a third-party parser.

Cross-lingual studies, common in typology and translation studies, are STUnD's main area of application. The first prototype of the tool (Masciolini & Tóth 2024) was in fact developed to facilitate the task of investigating the ways in which Swedish verb constructions are rendered in other more or less closely related languages. UD makes it possible to – at least to some extent – identify cross-lingual semantic correspondences between sub-sentence units on the basis of syntactic information alone (Masciolini & Ranta 2021) and allows queries to be largely language-agnostic. Parallel treebanks, however, need not necessarily be multilingual. We therefore propose two more use cases for STUnD: learner corpus research and quality control for linguistic annotation. In the former scenario, we suggest that the tool be applied to parallel *L1-L2 treebanks* (Lee, Li, et al. 2017) – a format increasingly popular within the UD community – for tasks such as grammatical error retrieval and analysis. As for the latter case, we show how the tool can be used to locate and examine discrepancies between different analyses of the same text. Together with STUnD's built-in editing functionalities, this can prove useful for resolving inter-annotator conflicts in the context of a treebanking project, but also play a role in parser evaluation, where system output often needs be compared against gold standard data or alternative automatic analyses.

This chapter is structured as follows. In Section 2, we provide a short introduction to the UD framework, followed by an overview of the treebank tools that most closely relate to STUnD (Section 3). Section 4 consists in a technical description of the tool itself, mostly intended as a practical introduction for new users but also detailing its inner workings. In Section 5, we present a multilingual case study which employs STUnD to explore the

3 Unlike in English, verbs in Swedish only have one present tense form. Without any further context, in fact, it is impossible to tell whether the Swedish sentence *jag lyssnar på musik* means 'I (usually) listen to music' or 'I am (right now) listening to music'.

tense-aspect forms used to render the Swedish *presens perfekt* tense into English and Japanese, while two additional use cases for STUnD are presented in Section 6. Finally, Section 7 is dedicated to some considerations about the further development of the tool and its application in future projects.

## 2 *Universal Dependencies*

Universal Dependencies (UD) (de Marneffe et al. 2021) is a well-established framework for dependency-based morphosyntactic annotation and a community effort which has resulted in a large collection of treebanks in over 150 languages, with applications in both linguistics and Natural Language Processing (NLP). As the term "Universal" suggests, the UD annotation scheme aims to maximize cross-linguistic consistency without sacrificing accuracy for the analysis of individual languages. This is done by providing a universal inventory of categories (part-of-speech tags, dependency relations and morphological features) and annotation guidelines while also allowing language-specific extensions. An exhaustive description of the UD standard is beyond the scope of this chapter.[4] We will, however, cover some of the basic principles of UD and introduce the text-based data format used to represent treebanks, both of which are necessary prerequisites for understanding the way STUnD works and the examples provided in the rest of this chapter.

### 2.1 *Basic principles*

UD annotation can be seen as a three-step process consisting of (1) sentence and word *segmentation*, (2) *word-level annotation* and (3) *syntactic annotation*. This section gives an example-based overview of each of these steps and introduces some commonplace UD terminology.

#### 2.1.1 *Segmentation*

UD analysis is carried out at the level of individual sentences, which are further *tokenized*, i.e. segmented into *syntactic words*. Tokenization can be a nontrivial problem, especially for languages with no orthographic word boundaries such as Japanese (see Section 5.2). For the sake of simplicity, however, we will momentarily assume an identity between orthographic (i.e. space-separated) and syntactic words. This largely holds for English and

4 For up-to-date annotation guidelines, we encourage the reader to visit the official UD website, universaldependencies.org.

Swedish, which will be used for all examples in this section, as well as for a wide variety of UD languages. It must be noted, however, that punctuation marks are considered as separate "words". Segmenting the sentence *We are annotating our first sentence.*, for example, results in the following seven tokens:

(1) We are annotating our first sentence .

### 2.1.2 *Word-level annotation*

Once a sentence has been tokenized, each word is *lemmatized* and assigned a coarse-grained or *Universal* Part-Of-Speech (UPOS) tag.[5] The words that make up our example sentence (1), for instance, can be analyzed as

| | PRON | AUX | VERB | PRON | ADJ | NOUN | PUNCT |
|---|---|---|---|---|---|---|---|
| (2) | We | are | annotating | our | first | sentence | . |
| | we | be | annotate | we | first | sentence | . |

In addition, each word can optionally be assigned a finer-grained, language-specific POS tag and a set of morphological features. A more complete annotation of the word *We*, for instance, would include the fine-grained POS tag `PRP` (personal pronoun) and the features `Case=Nom|Number=Plur|Person=1|PronType=Prs`, describing it as the nominative form of a first-person plural personal pronoun.[6]

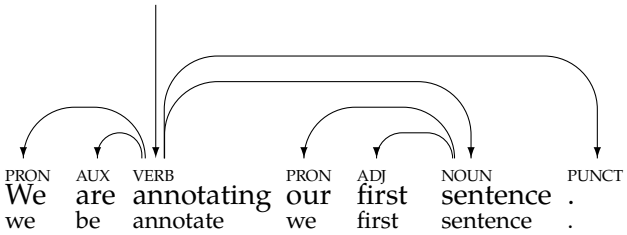
### 2.1.3 *Syntactic annotation – dependencies*

UD belongs to a class of grammatical theories known as *dependency grammar*, whose core idea is that structural information about a sentence can be captured by directed links, referred to as *dependencies*, between its parts.

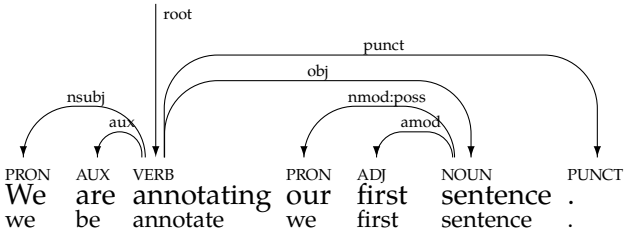
As in several other dependency-based frameworks, syntactic annotation in UD consists in establishing a set of dependency relations between individual words. These are one-to-one correspondences that assign each token a syntactic *head*. Nominals, for instance, are often the syntactic heads of the adjectives that modify them. Conversely, verbs typically have a subject, an object and possibly other complements as *dependents*. As illustrated below, these head-dependent links, represented by arrows, result in a tree structure rooted in one of the words (typically, but not always, the main verb):

5 The complete list of UPOS tags is available at universaldependencies.org/u/pos.

6 A list of all morphological features and their possible values is available at universaldependencies.org/u/feat/all.

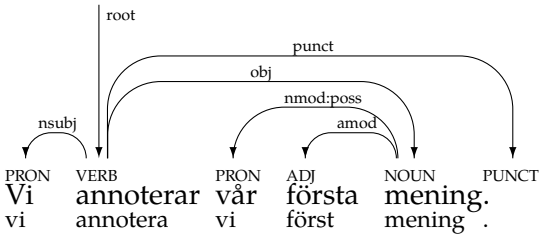
(3)



In UD, each link is also assigned a label that specifies the syntactic role of the dependent with respect to its head, i.e. the *relation type*:

(4)



In example (4) above, for instance, the `nsubj` label between *We* and the root of the tree, the verb *annotating*, specifies that *We* is the nominal subject of the sentence, whereas its object is the subtree labeled as `obj`, rooted in the noun *sentence*. The latter, in turn, has two dependents: the pronoun *our*, acting as a possessive nominal modifier – hence the label `nmod:poss`, where `nmod` is the main *relation type* and `poss` its *subtype* – and the adjectival modifier (`amod`) *first*.[7]

This first example also illustrates one of the core principles behind the UD annotation guidelines, that of *primacy of content words*, which states that dependency relations hold primarily between content words, while function words are usually attached as direct dependents of the most closely related content word. Unlike in most other dependency frameworks, in fact, the `root` of the sentence is not the auxiliary *are*, but the lexical verb *annotating*. The rationale behind this is to maximize cross-lingual parallelism. A straightforward Swedish translation of our example sentence, *Vi annoterar vår första mening.*, in fact, makes no use of auxiliaries but presents the exact same high-level structure:

(5)



7 For a complete taxonomy of UD relation labels, see universaldependencies.org/u/dep.

### 2.2 *Data format*

UD treebanks are distributed in a machine-readable text-based format called CoNLL-U.[8] A CoNLL-U file consists of a series of blocks, each representing a sentence. Sentence blocks are separated from each other by a blank line and can be preceded by metadata information, stored in lines starting with a hash sign (`#`). Metadata usually includes a sentence identifier (sent_id) and the unannotated text (`text`).

The words that make up a sentence are stored *vertically*, i.e. with one token and the annotations associated with it per line. Each line consists of a series of tab-separated values:

1. ID: progressive token index;
2. FORM: word form or punctuation symbol;
3. LEMMA: lemma or stem of the word form;
4. UPOS: universal POS tag;
5. XPOS: language-specific POS/morphological tag (optional);
6. FEATS: morphological features, stored as a list of key-value pairs in the format `Key1=Value1|Key2=Value2...` (optional);
7. HEAD: ID of the syntactic head of the current word (`0` for the root of the sentence);
8. DEPREL: relation type;
9. DEPS: enhanced dependency graph, stored as a list of head-deprel pairs (optional);
10. MISC: any other annotation, stored as a list of key-value pairs (optional).

Underscores (_) are used as placeholders for missing values. The CoNNL-U representation of the example sentence in (1–4) is given in Figure 1. For the sake of readability, we also provide a tabular representation of the same sentence (Figure 2).

## 3 *Treebank tools*

The success of UD as an annotation standard has led to a growing demand for tools and query languages for treebanks. In this context, a variety of

8 The CoNLL-U format is based on the format that was used in the context of the shared task on multilingual dependency parsing at the Tenth Conference on Computational Natural Language Learning (CoNLL-X).

```
# sent_id = ex1_en
# text = We are annotating our first sentence.
1     We          we          PRON   PRP    Case=Nom..   3     nsubj _     _
2     are         be          AUX    VBP    Mood=Ind...  3     aux   _     _
3     annotating  annotate     VERB   VBG    Tense=Pres... 0    root  _     _
4     our         we          PRON   PRP$   Case=Gen...   6     nmod:poss   _     _
5     first       first       ADJ    JJ     Degree=Pos... 6     amod  _     _
6     sentence    sentence    NOUN   NN     Number=Sing  3     obj   _     _
7     .           .           PUNCT  .      _            3     punct _     _
```

Figure 1: Textual representation of the example sentence in (1–4) in CoNLL-U format. For the sake of compactness, we omit some of the morphological features, as well as the DEPS and MISC fields.

| ID | FORM | LEMMA | UPOS | XPOS | FEATS | HEAD | DEPREL | DEPS | MISC |
|---|---|---|---|---|---|---|---|---|---|
| 1 | We | we | PRON | PRP | Case=Nom... | 3 | nsubj | _ | _ |
| 2 | are | be | AUX | VBP | Mood=Ind... | 3 | aux | _ | _ |
| 3 | annotating | annotate | VERB | VBG | Tense=Pres... | 0 | root | _ | _ |
| 4 | our | we | PRON | PRP$ | Case=Gen... | 6 | nmod:poss | _ | _ |
| 5 | first | first | ADJ | JJ | Degree=Pos... | 6 | amod | _ | _ |
| 6 | sentence | sentence | NOUN | NN | Number=Sing | 3 | obj | _ | _ |
| 7 | . | . | PUNCT | . | _ | 3 | punct | _ | _ |

Figure 2: Tabular representation of the CoNNL-U fragment shown in Figure 1, representing the example sentence in (1–4). For the sake of compactness, we omit some of the morphological features, as well as the DEPS and MISC fields.

services such as PML-TQ (Prague Markup Language – Tree Query, Pajas & Štěpánek 2009), SETS (Scalable and Efficient Tree Search, Luotolahti et al. 2015) and Grew-match (Guillaume 2021) were extended and/or developed from scratch to work with CoNLL-U files.

What many of these tools have in common is that, as opposed to classic string- or sequence-based corpus search tools, they allow searching for syntactic constructions based on the structural information present in CoNLL-U files (cf. Section 2.1). In Grew syntax, for instance, the query

```
pattern {
    X [VerbForm = Part];
    Y [lemma = "have", Tense = Pres];
    X -> Y;
}
```

can be used to search for occurrences of the English *present perfect* (*have* + past participle). To do that, it specifies that results must contain at least two tokens `X` and `Y`, respectively a participle and a present form of the verb *to have*, and that the former must be a dependent of the latter.[9] STUnD uses a

9 This query can be tested on official UD treebanks at `universal.grew.fr`.

different query language, largely based on the one used in the gf-ud toolset for dependency trees and interlingual syntax (Kolachina & Ranta 2016, Ranta & Kolachina 2017), but is more or less equivalent to Grew-match in terms of expressivity of the queries, at least when used on a single treebank.

When it comes to parallel data, however, the number of available options is much more limited. Some tools, such as Grew-match itself, allow displaying hits alongside the corresponding sentence in a parallel treebank based on sentence-level metadata. However, none of the existing query engines aside from STUnD's command-line predecessor, L2-UD (Masciolini 2023), allow performing parallel searches such as those discussed in the introduction.

This is not to say that STUnD is the only tool with a focus on treebank comparison. The SETS-based treebank analysis tool STARK (Krsnik et al. 2019), for example, allows extracting comparative statistics and dependency patterns from pairs of UD treebanks (not necessarily parallel). The comparisons enabled by STARK, however, are treebank-level rather than sentence-level. Some annotation platforms such as Arborator-Grew (Guibon et al. 2020), on the other hand, offer the possibility to compare alternative analyses of the same text by highlighting annotation differences between individual words. To the best of our knowledge, however, none of the existing tools allow sentence-level comparisons between a text and its translation or two otherwise different versions of a given text.

## 4 *Tool description*

In the following, we provide a technical description of STUnD, aimed for both new users and programmers interested in contributing to the tool's further development or in reusing some of its components. Implementation details intended for the latter group are enclosed in grey text boxes that can be safely skipped by readers whose main focus is learning how to use the finished product.[10] We start with a presentation of the tool's user interface (Section 4.1) and input formats (4.2), followed by a discussion of the underlying query engine and language (4.3). Section 4.4 introduces some additional features.

### 4.1 *User interface*

STUnD is a web-based treebank search application consisting of a Haskell backend and a simple HTML+JavaScript frontend. The software is open

10 A step-by-step tutorial is also available at harisont.github.io/STUnD/tutorial.

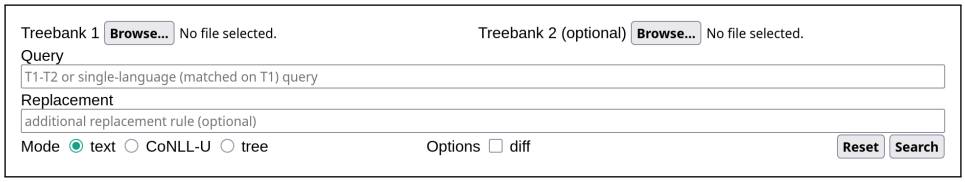



Figure 3: STUnD's input fields.

source[11] and easy to run locally.[12] In addition, a live demo of the tool is available at harisont.github.io/STUnD.

Upon launching STUnD, the user is presented with a series of input fields (cf. Figure 3). Two file selectors ("Treebank 1" and "Treebank 2") allow choosing one or two input files, henceforth $T_1$ and $T_2$. These can be UD treebanks in CoNLL-U format or plain text files (cf. Section 4.2). Through the two text input fields, "Query" and "Replacement", the user can specify a search and a replacement pattern, both discussed in depth in Section 4.3.[13] Two buttons, "Search" and "Reset" are used to run the query and clear the input fields respectively.

Before running a query, the user can choose in what "Mode" to get the search results. In "text" mode (default, cf. Figure 4), hits are presented in a format similar to KeyWords In Context (KWIC) or concordances, i.e. highlighted in bold in the context of the full sentences in which they appear. Results can be downloaded as plain text or, when a parallel treebank is searched, TSV (Tab-Separated Values) by clicking on "parallel file". This is intended for further processing, for instance in a spreadsheet program.

In "CoNLL-U" mode (cf. Figure 5), results can be inspected in their underlying structured format. Unlike text mode, CoNLL-U mode isolates the sentence fragments matching the query. Results consist of well-formed UD trees , which can be downloaded by clicking on "T1 file" and "T2 file" and fed back into STUnD for finer-grained corpus analysis. Alternatively, they can be stored as a single treebank – where $T_1$ and $T_2$ sentences are interleaved – by clicking on "parallel file".

Finally, query-matching segments can be rendered as graphical UD trees (cf. Figure 6). Exporting is similar to CoNLL-U mode, but the resulting files consist of a series of SVG (Scalable Vector Graphics) fragments, easy to embed in a webpage or directly visualize in a web browser.

11 The source code can be found at github.com/harisont/STUnD.
12 See installation instructions at harisont.github.io/STUnD/installation.
13 Note that these fields are both optional: by leaving them empty, the user can inspect the full treebank.

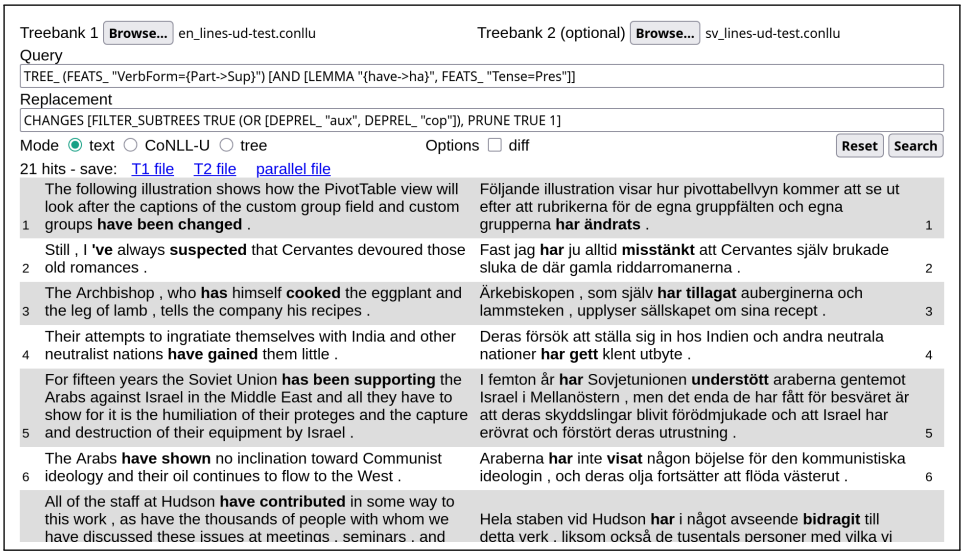



Figure 4: Using STUnD to look for sentences containing occurrences of literally translated present perfect in the English-Swedish parallel treebank LinES (Ahrenberg 2007, 2015) in text mode. A replacement pattern is used to isolate the relevant verb constructions.

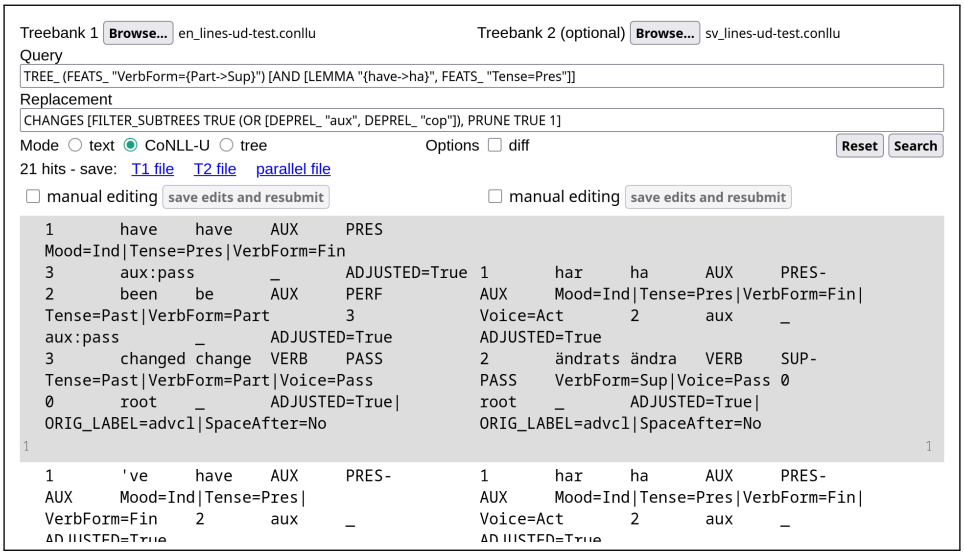



Figure 5: Search results for the same query as in Figure 4 in CoNLL-U mode.

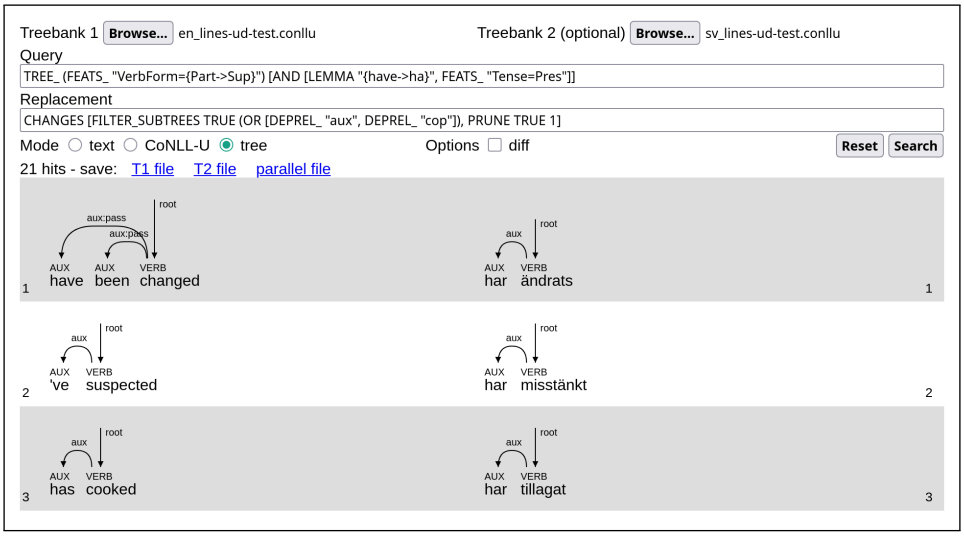



Figure 6: Search results for the same query as in Figure 4 in tree mode.

## 4.2 *Input formats*

STUnD's first prototype only accepted CoNLL-U input. The number of available pre-annotated parallel UD treebanks is, however, rather limited. Therefore, users often need to automatically process other texts with a dependency parser. To make our tool more accessible, we opted for integrating parsing into STUnD itself, which now also supports raw text input.

If, upon clicking on "Search", one or both input files consist of raw text, the first step is determining their language. If the file name starts with a two letter ISO 639-1 language code[14] followed by an underscore (e.g. `sv_` for Swedish), the text is assumed to be in the corresponding language. Otherwise, the language of the text is automatically inferred.

Once the language has been determined, after asking the user for consent, the text is sent to the UDPipe API,[15] where it is parsed with the default model for the language in question. If parsing is successful, the given query is run directly on the output. Afterwards, however, any parse errors can be manually corrected with STUnD's manual editing functionality (cf. Section 4.4.3).

14 See `loc.gov/standards/iso639-2/php/English_list.php` for the full list of codes.
15 `lindat.mff.cuni.cz/services/udpipe/api-reference.php`

*Under the hood: Language identification* Language identification is based on `langid.js`,[a] a JavaScript implementation of `langid.py` (Lui & Baldwin 2012) which uses a multinomial Naïve Bayes classifier over byte $n$-grams (with $1 \leq n \leq 4$). The model has been trained on data for 97 languages and 5 domains: government documents, software documentation, newswire, Wikipedia, and internet crawls. It has been shown that this method achieves high accuracy across domains (Lui & Baldwin 2011).

*a* github.com/saffsd/langid.js

## 4.3 *STUnD queries*

This section delves into the core of STUnD: the query engine. We start by discussing how the tool can be used to explore individual treebanks, presenting the basic query language and pattern matching algorithm (Section 4.3.1). We then move on to querying parallel treebanks, discussed in Section 4.3.2. Finally, Section 4.3.3 introduces replacement patterns, which allow applying systematic changes to the search results so as to refine them.

### 4.3.1 *Simple queries*

STUnD queries are based on the pattern matching functionalities of the gf-ud toolset for dependency trees and interlingual syntax (Kolachina & Ranta 2016, Ranta & Kolachina 2017),[16] whose UD-related utilities are now maintained as a separate software library, deptreehs.[17] The latter provides a Haskell-embedded pattern matching language with a concise syntax especially well-suited for describing syntactic (tree) structures.

In the following, the aim is to give the reader a good understanding of the fundamental constructs of the query language so as to be able to follow the examples presented in this chapter and get started with writing their own queries. Certain details are therefore omitted. The full specification of the query language is available as part of the deptreehs documentation.[18]

*Basic patterns* The most basic STUnD patterns describe the contents of a single CoNLL-U field. They consist of the name of the field itself followed by a string of text, delimited by double quotes, that specifies its desired value. The currently supported fields are FORM, LEMMA, UPOS, XPOS, FEATS, DEPREL and MISC. When a query consists of a single basic pattern,

16 Source code available at github.com/GrammaticalFramework/gf-ud.
17 Source code available at github.com/harisont/deptreehs.
18 github.com/harisont/deptreehs/tree/main/docs

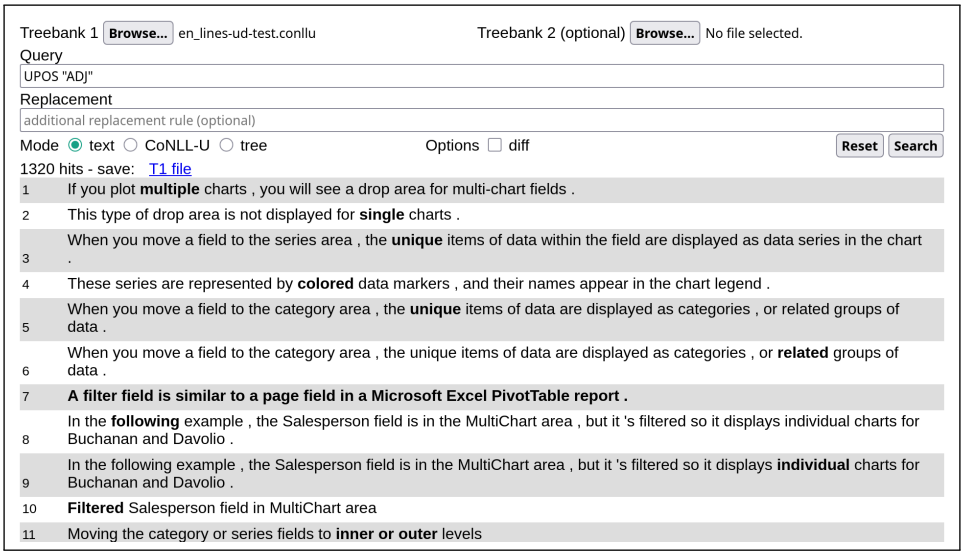



Figure 7: Search results for the query `UPOS "ADJ"` on the English half of the LinES treebank. Most hits consist of single tokens, but when adjectives have a modifier or a conjunct, such as in hit number 11, the entire subtree is retrieved. In one case, the adjective *similar*, used as a copula complement, is the root of the sentence, which is therefore considered a match in its entirety (cf. hit number 7).

such as `UPOS "ADJ"`, STUnD will retrieve all dependency subtrees rooted in an adjective encountered in the corpus, regardless of their "depth" in their parent tree (cf. Figure 7).

Strings used in a basic pattern can start or end with the wildcard `*`, matching any substring. The pattern `DEPREL "*comp"`, for example, matches both classes of UD clausal complements (`ccomp` and `xcomp`). In addition, some basic patterns have a "relaxed" version, characterized by an underscore following the CoNLL-U field name. The pattern `DEPREL_ "aux"`, for instance, ignores relation subtypes, matching subtrees labelled both `aux` and `aux:pass`, whereas its stricter counterpart `DEPREL "aux"` only matches auxiliaries in non-passive constructions.

*Compound patterns* Basic patterns can be combined by means of *logical operators*, into *sequences* and/or into *tree patterns*.

The query language provides three logical operators: `AND`, `OR` and `NOT`. For example, the pattern

```
AND [OR [UPOS "VERB", UPOS "AUX"], NOT (DEPREL "cop")]
```

matches all (sub)trees rooted in a verb or auxiliary, except those used as a

copula. As displayed in the example above, the `AND` and `OR` operators are followed by a comma-separated list of patterns delimited by square brackets, while the `NOT` operator is followed by a single pattern, enclosed in parentheses.[19] Additionally, the pattern `TRUE` can be used to match any subtree.

Similar to `AND` and `OR` expressions, sequences of patterns can be built by prepending the keyword `SEQUENCE` to a list of patterns. The pattern

```
SEQUENCE [DEPREL "nmod:poss", UPOS "NOUN"]
```

for instance, matches all possessives immediately followed by a noun. If the sequence need not be contiguous, a `SEQUENCE_` pattern, where this constraint is relaxed, can be used instead.

Last but not least, tree patterns, used to search for dependency-syntactic structures, are the real strength of the query language. The form of a tree pattern is `TREE p [p1, p2, p3...]`, where `p` is a pattern to be matched by the head of the subtree, while `p1`, `p2`, `p3`... are to be matched by its dependents in the order in which they appear in the query, without any other intervening subtrees. For example, the pattern

```
TREE (UPOS "NOUN") [DEPREL "nmod:poss"]
```

appears similar to the previous example, but there are some important differences between the two. First of all, the tree pattern specifies that there has to be a head-dependent relation between the noun and its modifier, i.e. a link that connects the former to the latter. Therefore, search results with this query will include nouns that have a possessive modifier attached to them, independent of the order in which the two elements appear and even if there are other intervening words. This pattern, however, excludes all trees that have both a determiner and some other token among their dependents.

A more relaxed – and more commonly used – version of this pattern, `TREE_`, allows for additional dependents at any point in the sequence. Replacing `TREE` with `TREE_` in the pattern above therefore produces a query that matches all noun-possessive pairs in a head-dependent relation, including those with other modifiers. The running example from Section 2.1.3, *We are annotating **our** first **sentence**.* – whose dependency analysis is given in (4) – would only be matched by this second version of the tree pattern, since the word *first* occurs between the possessive pronoun and the noun and is also a dependent of the latter.

Finally, the slightly more complex pattern

```
TREE_ (FEATS_ "VerbForm=Part") [AND [LEMMA "have", FEATS_
    "Tense=Pres"]]
```

19 Note that square brackets are used to delimit *lists* of patterns, whereas round brackets surround single patterns whenever they occur as part of larger compound ones, similar to parentheses in a mathematical expression.

matches all participles with a present-tense inflection of the verb *to have* among its dependents, i.e. all English present perfect constructions.[20] Its non-underscored version, on the other hand, would only retrieve instances where *have*/*has* is the only dependent of the participle.[21]

*Under the hood: Basic query matching* From a technical standpoint, when a query is run on a treebank $T$, STUnD goes through all sentences in the treebank and visits each tree $t$ and all of its subtrees recursively, looking for any subtree $t'$ matching the given pattern. Formally, search results always consist of a set of subtrees (possibly several per sentence), with two edge cases: (1) single tokens, i.e. "leaves" and (2) complete sentences, i.e. full dependency trees. This means that, in all cases excepts the latter, results may not be well-formed CoNLL-U trees with an explicit root and progressive token IDs starting from 1. STUnD, however, rescales the ID and HEAD fields and overwrites the dependency label of the head of each extracted subtree with `root`, storing its original value in the MISC field. This is to ensure that search results can be further processed without any post-editing.

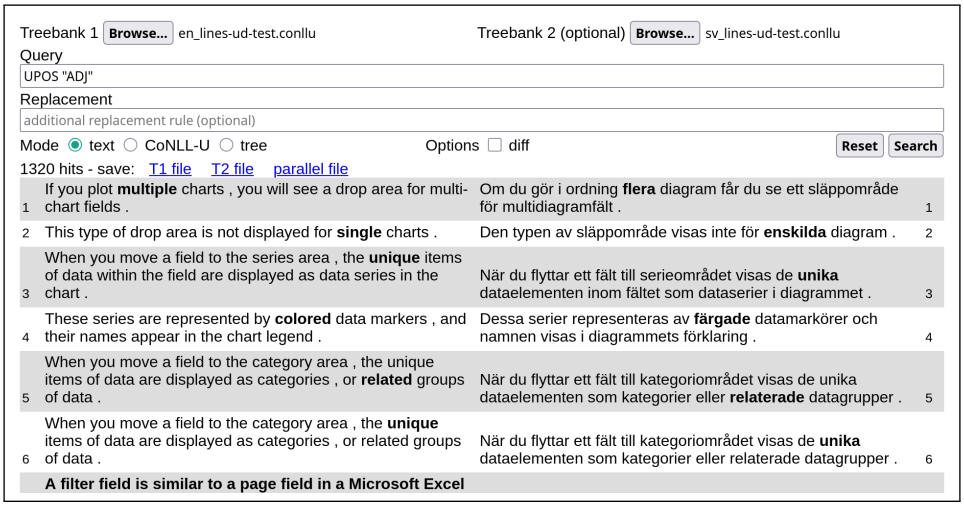
### 4.3.2 *Parallel queries*

Up to this point, we have discussed how simple queries can be applied to a single UD treebank. This is similar to what can be achieved with all treebank search tools mentioned in Section 3. STUnD, however, was designed with the goal of facilitating parallel treebank exploration, which will be the focus of this section. We start by discussing what happens when a query is run on two sentence-aligned treebanks and then move on to query language extensions specifically meant for parallel search.

When a query is run on a parallel corpus, STUnD matches it on the left-hand-side treebank ($T_1$) as described in Section 4.3.1, but also attempts to find a semantic correspondence in the right-hand-side treebank ($T_2$). If, as in Figure 8, the user looks for adjectives, hits will consist of subtree pairs consisting of an element of $T_1$ that matches the query directly (i.e. is rooted in an adjective) and its counterpart in $T_2$. The latter does not necessarily need to match the query: in hit number 7, for instance, the entire sentence is rooted in the adjective *similar*, used as a copula complement for the auxiliary *is*, whereas the root of the Swedish translation is the verb *liknar* 'resemble'.

20 This pattern is equivalent to the Grew-match query shown in Section 3.

21 This is never the case in standard English because the obligatory subject of a present perfect-tense clause is also always a dependent of its head, which is the participle itself.



Treebank 1 Browse... en_lines-ud-test.conllu  Treebank 2 (optional) Browse... sv_lines-ud-test.conllu

Query

UPOS "ADJ"

Replacement

additional replacement rule (optional)

Mode ◉ text ○ CoNLL-U ○ tree  Options ☐ diff  Reset Search

1320 hits - save: T1 file T2 file parallel file

| | | | |
|---|---|---|---|
| 1 | If you plot **multiple** charts , you will see a drop area for multi-chart fields . | Om du gör i ordning **flera** diagram får du se ett släppområde för multidiagramfält . | 1 |
| 2 | This type of drop area is not displayed for **single** charts . | Den typen av släppområde visas inte för **enskilda** diagram . | 2 |
| 3 | When you move a field to the series area , the **unique** items of data within the field are displayed as data series in the chart . | När du flyttar ett fält till serieområdet visas de **unika** dataelementen inom fältet som dataserier i diagrammet . | 3 |
| 4 | These series are represented by **colored** data markers , and their names appear in the chart legend . | Dessa serier representeras av **färgade** datamarkörer och namnen visas i diagrammets förklaring . | 4 |
| 5 | When you move a field to the category area , the unique items of data are displayed as categories , or **related** groups of data . | När du flyttar ett fält till kategoriområdet visas de unika dataelementen som kategorier eller **relaterade** datagrupper . | 5 |
| 6 | When you move a field to the category area , the **unique** items of data are displayed as categories , or related groups of data . | När du flyttar ett fält till kategoriområdet visas de **unika** dataelementen som kategorier eller relaterade datagrupper . | 6 |
| | **A filter field is similar to a page field in a Microsoft Excel** | | |

Figure 8: Search results for the query `UPOS "ADJ"` on the parallel English-Swedish treebank LinES.

For finding such semantic correspondences, STUnD relies on the concept-alignment Haskell library[22] (Masciolini & Ranta 2021), which uses a set of customizable rules to determine which sentence fragments correspond to each other in a parallel treebank. This software package was originally designed for automatically bootstrapping translation lexica from multilingual corpora, but can also be applied, sometimes with even better results, to other types of parallel data (cf. Section 6).

*Under the hood: Subtree alignment* Technically, what concept-alignment does is comparing two sentence-aligned UD treebanks and recursively extracting pairs of semantically corresponding subtrees, i.e. words or phrases, based on a set of customizable *alignment criteria*. In the case of STUnD, these are heuristics based on UD's cross-lingual abstractions, such as comparing dependency labels, UPOS tags of content words and – for same-language parallel treebanks – lemmas. It follows that the quality of the alignments varies depending on the syntactic similarity between the two treebanks and on the quality of the UD annotation. For a manually annotated corpus of Swedish to Norwegian translations, for instance, concept-alignment is quite reliable. At the opposite extreme, results obtained from an automatically parsed corpus of texts in two typologically unrelated languages can be less satisfactory, especially if tokenization conventions differ. For this reason, it is important for matches to also be available in the context of the full sentences they appear in.

22 github.com/harisont/concept-alignment.

*Under the hood: Matching queries on parallel treebanks* The parallel query matching algorithm, first introduced in Masciolini (2023) is as follows:

1. the alignment module extracts phrase- and word-level alignments from the input parallel treebank $\langle T_1, T_2 \rangle$. The result of this step can be seen as a larger, intermediate parallel treebank $\langle T'_1, T'_2 \rangle$ where each element is a pair of aligned UD subtrees retaining information about the "parent sentence" of each subtree
2. $T_1$ is recursively scanned for subtrees matching the query, exactly as discussed in Section 4.3.1
3. each matching subtree $t_1$ is looked up in $T'_1$. At this point, there are two possibilities:
   - if it is found, the hit will consist of the subtree alignment $\langle t'_1, t'_2 \rangle$, where $t_1 = t'_1$
   - if there is no such subtree pair in $\langle T'_1, T'_2 \rangle$, it means that the alignment step has failed to find a counterpart for $t_1$ in $T_2$[a] and $t_1$ is returned alongside a "dummy" empty UD tree. STUnD's UI, however, still makes it possible – at least in text mode – to compare the $\langle t_1, t_2 \rangle$ full sentence pair the match occurs in.

*a* This is sometimes due to the limitations of the alignment module, but also common and expected when the query looks for something eminently language-specific, such as certain function words. Looking for definite determiners in an English-Swedish treebank, for instance, typically results in a lot of partial matches, since definiteness is mostly expressed morphologically in Swedish.

*Divergence patterns* We have seen how basic queries alone can be a useful tool for searching parallel treebanks. In a number of cases, however, users might want to specify what exactly to look for in each of the two halves of a parallel treebank. When studying translation divergences, for instance, it may be useful to retrieve all instances where a direct object in the source text is rendered as an oblique in its translation (cf. Figure 9) rather than looking for all English objects and their Swedish counterparts only to filter the results manually.

In STUnD, this is possible through so-called *divergence patterns*, i.e. queries containing a $T_1$ and a $T_2$-specific portion. A divergence pattern essentially consists of two basic queries to be matched in parallel, one on $T_1$ and one on $T_2$. Looking for objects translated as obliques, for instance, means looking for subtree alignments where $t'_1$ is labelled as `obj` and $t'_2$ as `obl`. For the sake of conciseness, however, the basic query language has been extended with an arrow syntax which allows specifying the treebank-specific portions of

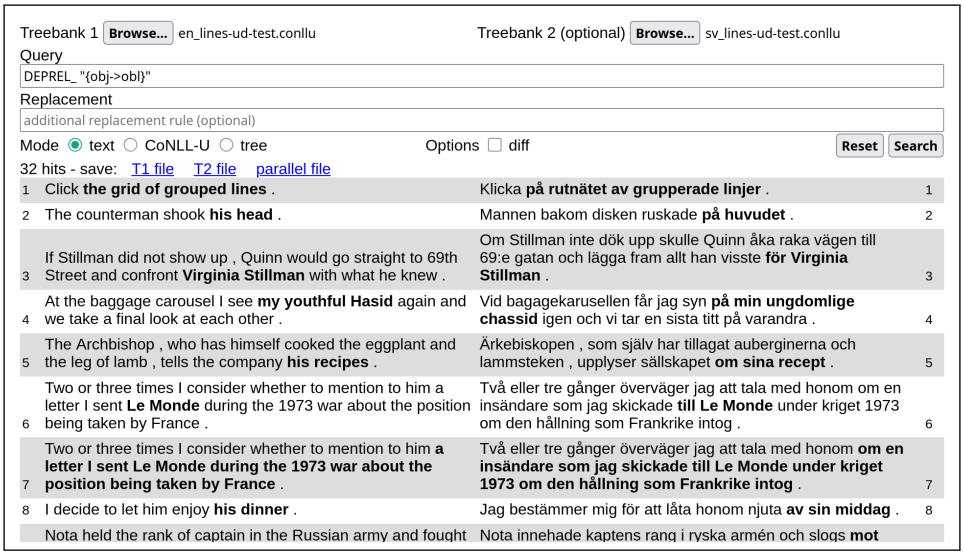



Figure 9: STUnD used to search for translation divergences in the parallel English-Swedish treebank LinES.

the query without rewriting the entire pattern twice. The query

```
DEPREL_ "{obj -> obl}"
```

for example, reads as "find all $T_1$ subtrees whose head is a direct object and whose counterpart in $T_2$ is a subtree rooted in an oblique".

*Under the hood: Matching divergence patterns* The matching algorithm for divergence patterns is exactly the same used when running a basic query on a parallel treebank, except that the arrow notation is expanded into two separate queries $\langle q_1, q_2 \rangle$ (in this simple example, $q_1$: `DEPREL_ "obj"` and $q_1$: `DEPREL_ "obl"` and that a sentence pair $\langle t_1, t_2 \rangle$ alignment is only considered a match if, given one of its $\langle t'_1, t'_2 \rangle$ alignments, $t'_1$ matches $q_1$ and $t'_2$ matches $q_2$.

### 4.3.3 *Replacement patterns*

As mentioned previously, STUnD offers replacement patterns as a way to refine search results. More broadly, replacement patterns can be used to apply arbitrary changes to a treebank. This can be useful in settings other than treebank querying, e.g. for fixing systematic annotation errors or adapting an existing treebank to updated annotation guidelines that affect specific linguistic phenomena.


Figure 10: STUnD used to search for occurrences of the Swedish *presens perfekt*, with and without using a suitable replacement pattern. On the right hand side, the replacement is used to isolate the verb construction.

The replacement patterns most widely used in STUnD, however, are the ones that allow filtering or pruning the trees that the user obtains as a result of running a query, thus isolating their most salient parts.

```
FILTER_SUBTREES (UPOS "VERB") (UPOS "AUX")
```

for instance, can be used to remove all dependents of verbs other than their auxiliaries. The query

```
PRUNE TRUE 1
```

on the other hand, decreases the depth of all subtrees to 1, thus reducing the search results to the subtrees' heads and their immediate dependents.

It is also possible to combine several changes into a single replacement pattern. There are two ways to do this. Prepending the keyword `CHANGES` to a list of replacements `[X,Y,Z]` goes through all subtrees looking for possibilities to perform each operation and applies the first applicable one. With `COMPOSE [X,Y,Z]`, on the other hand, the edit operation `X` is performed first wherever possible, `Y` is applied to the resulting treebank, `Z` on the results of `Y` and so on. For example,

```
COMPOSE [FILTER_SUBTREES (UPOS "VERB") (UPOS "AUX"), PRUNE TRUE 1]
```

combines the two patterns discussed above making it so that pruning happens only after irrelevant subtrees have been filtered out (for a real-world example of application of this pattern, see Figure 10).

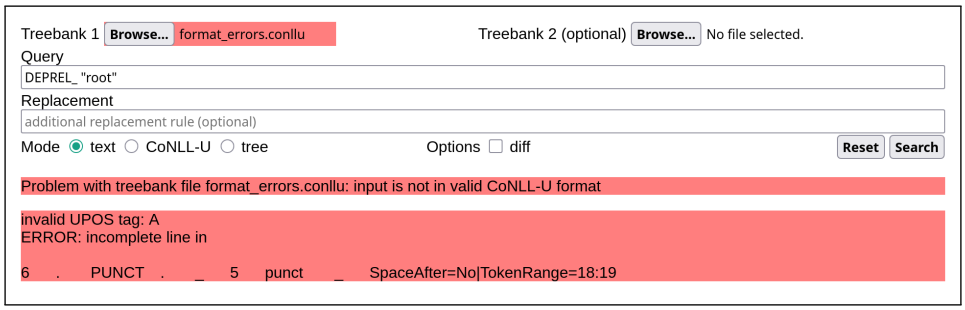



Figure 11: Trying to load an invalid CoNNL-U file into STUnD. The results are two error messages, one signaling that `A` is not a valid Universal POS tag, one indicating that one of the lines does not contain all expected values.

## 4.4 *Other functionalities*

Aside from treebank querying, STUnD has a set of features geared towards treebank annotators: treebank validation, diff mode and manual editing.

### 4.4.1 *Validation*

In the examples above, STUnD was used to explore officially released treebanks, the gold standard for UD annotation. Treebanks under development, however, often present format errors that can hinder or completely prevent processing with STUnD, leading to issues such as failures to render dependency trees and misleading query results. For this reason, STUnD performs some basic validation of the input data, checking that all word lines contain 10 tab-separated fields, as well as that all token IDs, UPOS tags and DEPRELs are valid as per the UD universal annotation guidelines. If the treebank contains one or more format errors, these are listed in the user interface (cf. Figure 11). This feature has proved to be useful when grading UD annotation assignments and can help new treebanks in the initial phases of their development, but it should not replace the official UD validator.[23]

### 4.4.2 *Diff mode*

Diff mode helps users identify discrepancies between two similar treebanks. This feature can be useful when exploring parallel L1-L2 treebanks, as well as when comparing alternative analyses of the same text. Both use cases will be

23 The official UD validator can be downloaded at github.com/UniversalDependencies/tools/blob/master/validate.py and is automatically run whenever an official UD repository is added or updated.

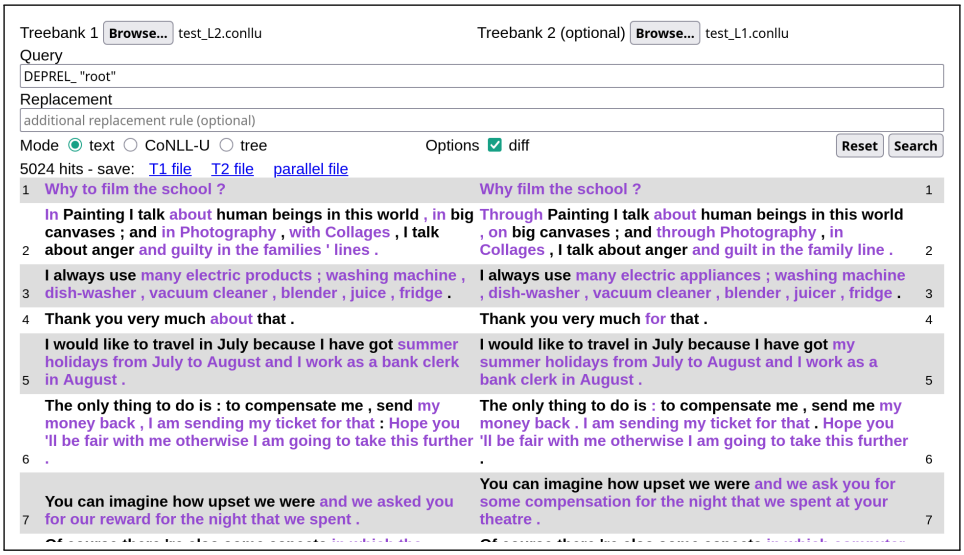



Figure 12: STUnD to browse the test set of the English as a Second Language (ESL) treebank (Berzak et al. 2016) with diff mode enabled. Treebank 1 consists of original learner productions, while Treebank 2 contains the corresponding corrections. Therefore, the highlighting helps identifying grammatical errors.

discussed in Section 6. With diff mode enabled, pairs of aligned subtrees that diverge in terms of surface word forms and/or annotation are highlighted in purple, as illustrated in Figure 12. Comparing subtrees rather than individual tokens yields visualizations that are sometimes coarser-grained compared to those available in text-based diff tools and annotation platforms such as Arborator-Grew (Guibon et al. 2020), but more flexible: it is in fact possible to use STUnD's diff mode on any two variants of a given text, potentially even a text and its translation to a different language, provided that the definition of "divergence" is adequately relaxed.[24]

### 4.4.3 *Manual editing*

In Section 4.3.3, we discussed how replacement patterns can be used not only to refine search results, but also to apply arbitrary changes to the treebank(s). This, however, is often not the most practical way to edit individual trees. The user can therefore also edit the treebank manually. This is currently supported in CoNLL-U mode only, with plans to add a table view for less

24 STUnD's UI does not currently make it possible to configure the tool to obtain meaningful diffs when comparing different languages, but the underlying code is easy to modify in this sense.

error-prone editing. Compared to other CoNLL-U editors such as Arborator-Grew (Guibon et al. 2020), in any case, STUnD's editing functionalities are basic. Editing in STUnD should therefore be seen as a quick way to apply the final touches to a treebank that is undergoing annotation rather than as the main annotation tool for a treebanking project in its early stages.

## 5 *Case study: cross-linguistic comparison*

In the following, we demonstrate the use of STUnD in a case study. The case study explores the usage of the Swedish and English present perfect tense-aspect forms and its Japanese equivalents. While Japanese has no tense-aspect form directly corresponding to the present perfect form of Swedish and English, we chose the form *-teiru* in Japanese as a nearly corresponding form, as *-teiru* has reportedly similar uses to the present perfect of English (Shirai 2000, Tóth 2022). For the sake of simplicity, we will use the abbreviation *PresPf* for the present perfect when referring to both the present perfect in English and its counterpart *presens perfekt* in Swedish. We use language-specific namings for the categories of Swedish and Japanese, in order to avoid confusion with English categories. The overarching purpose of the case study is to show how the STUnD tool can be used to make cross-linguistic observations on grammatical structures, more specifically on tense-aspect systems.

Languages have various grammatical forms to express time and by investigating the distribution of the forms cross-linguistically we can draw conclusions on the similarities and differences between tense-aspect systems of different languages. The PresPf is interesting in this sense since it shows great cross-linguistic variation (De Swart 2016). For instance, the use of the PresPf in Swedish interrogative clauses such as *Har du sovit gott?* is not unusual, while the preterite *Did you sleep well?* would be preferable in English in a similar situation (Larsson & Lyngfelt 2011). Thus, the questions are to what extent the use of the PresPf agrees between two or more languages, and what forms a language which has no designated PresPf form – such as Japanese – uses in the same situations.

### 5.1 *The tense-aspect systems of English, Swedish and Japanese*

Swedish and English have a similar division of tense forms, shown in Table 1, where we exemplify the tense forms with the verb *to run* in English and *springa* 'run' in Swedish. The Swedish tense forms correspond approximately to those of English, although these tense forms are used slightly differently in the two languages.[25]

Table 1: Tense forms in English and Swedish.

| English tense forms | Swedish tense forms |
|---|---|
| Simple present:<br>*runs* | Enkelt presens:<br>*springer* |
| Preterite:<br>*ran* | Preteritum:<br>*sprang* |
| Present perfect:<br>*has run* | Presens perfekt:<br>*har sprungit* |
| Pluperfect:<br>*had run* | Pluskvamperfekt:<br>*hade sprungit* |
| will + infinitival verb:<br>*will run* | ska + infinitival verb:<br>*ska springa* |
| would + infinitival verb:<br>*would run* | skulle + infinitival verb:<br>*skulle springa* |
| (is) going to + infintival verb:<br>*is going to run* | kommer att + infinitival verb:<br>*kommer att springa* |
| (was) going to + infinitival verb:<br>*was going to run* | kom att + infinitival verb:<br>*kom att springa* |

One point of difference between the two languages is that English has a form for progressive aspect, namely the *be* + *ing*-construction, which can be combined with the PresPf, as in *I have been eating* (cf. *I have eaten*; both forms can be translated to Swedish with the PresPf *jag har ätit*). Swedish has a repertoire of constructions for expressing progressive aspect (e.g. *hålla på att äta*, lit. 'hold on to eat') but it has no inflectional form for progressive aspect. The PresPf of Swedish consists of the auxiliary *ha* in present tense 'have' and the supine form[26] of the main verb, e.g. *har ätit* 'has eaten', and refers to a past event with some sort of relevance at present time (Teleman et al. 1999). A sentence such as *Jag har ätit min macka* 'I have eaten my sandwich' indicates that as a result of eating the sandwich in the past, the sandwich 'is' eaten at present. In Swedish, the auxiliary *ha* can be omitted in subordinate clauses e.g. *Jag ser att du redan* (*har*) *ätit* 'I see that you (have) already eaten', meaning that the supine form appears without an auxiliary. The Swedish

25 For instance, in cases where English would use the future tenses *will run* or *going to run*, the preferred counterpart in Swedish is often *enkelt presens* (the simple present): *springer* 'run(s)'.

26 The supine form (e.g. *brutit* 'broken') of Swedish, used in the present perfect, is distinct from the past participle (e.g. *brutet* 'broken').

Table 2: Tense-aspect forms in Japanese.

| | Non-past | Past |
|---|---|---|
| Non-continuous | *-ru* | *-ta* |
| Continuous | *-tei-ru* | *-tei-ta* |

PresPf corresponds to the English present perfect, which consists of the auxiliary *have* in present tense and the past participle, e.g. *have eaten*.

Japanese has four basic tense-aspects forms (Kudou 1995), as summarized in Table 2. The four forms are inflections and thus directly attach to verbs. All four forms convey both temporal (past/non-past) and aspectual (continuous/non-continuous) information. Non-past forms are generally used for events occurring at the present moment or in the future, while past forms are generally used for events occurring in the past. The continuous forms often express progressive aspect – however, see (7) below. The four forms are exemplified below.[27]

(6) a. *Ashita wa sushi wo tabe-ru*
Tomorrow TOP sushi ACC eat-NONPAST
'I will eat sushi tomorrow'

b. *Kinou wa sushi wo tabe-ta*
Yesterday TOP sushi ACC eat-PAST
'I ate sushi yesterday'

c. *Ima, sushi wo tabe-tei-ru*
Now sushi TOP eat-ASP-NONPAST
'I'm eating sushi right now'

d. *Sono toki, sushi wo tabe-tei-ta*
That time, sushi ACC eat-ASP-PAST
'At that time, I was eating sushi'

However, all four forms interact with lexical aspect and the aspectual interpretation of a form might depend on the verb used. For instance, the *-teiru* form tends to indicate stativity with telic verbs (verbs with an inner boundary), as in (7) below.

(7) *Terebi ga tsui-te-iru*
TV NOM turn.on-ASP-NONPAST
'The TV is turned on'

27 The list of abbreviations used in the glosses can be found in at the end of this chapter.

The default interpretation of (7) is that as a result of the TV being turned on, it 'is' turned on at the present moment. Furthermore, *-teiru* can also indicate that a past event has some sort of connection to the present, as in (8 a–b).[28]

(8) a. *Satou wa, 10 nen mae kara kono kaisha de hatarai-tei-ru*
Satou TOP 10 year before since this company LOC work-ASP-NONPAST
'Satou has been working at this company since ten years back'

b. *Mado, ake-tei-ru yo*
window open-ASP-NONPAST PART
'I've opened the window'

In (8 a), the event of 'working' has held from ten years back up to present time. In (8 b), the event of opening the window has a result at present, namely that the window is open.

Japanese has forms which are used for coupling sentences, similarly to conjunctions such as *and* in English. These forms are the verb stem (e.g. the verb *hashi-ru* 'to run' becomes *hashi-ri*) and the *-te*-form (e.g. *hashi-ru* → *hashi-tte*), as in *Kawa made hashi-tte/hashi-ri, ie ni kaetta* '(I) ran to the river, and returned home'. A written language variety of *-teiru*, namely *-teoru*, can also be inflected for the verb stem *-teori* when coupling sentences.

Morphologically speaking, the continuous forms in (6 c–d) consist of the *-te*-form followed by the auxiliary *i-ru*, which can be conjugated for past tense, i.e. *i-ta*. However, since *-teiru* and *-teita* function holistically as a verb conjugation, we will not divide the glossing of Japanese examples into *-te* and *i-ru*/*i-ta*, but will refer to these forms as *-teiru* and *-teita*.

Japanese has constructions that express tense-aspect, such as the aforementioned *-te*-form combined with the verb *kuru* 'to come' and *iku* 'to go', which can also be conjugated in the *-ta*-form, i.e. *ki-ta* 'came' and *it-ta* 'went'. For instance, *-te kuru* as in *fue-te kuru* means that the event ('increase' in this case) has been progressing 'up to a certain point in time'.

The *-ta koto ga aru*-construction is a sort of experiential construction and expresses that the person or thing referred to has the prior experience of participating in the event in question. Thus, *Igirisu ni i-tta koto ga a-ru* means that '(I) have been to the U.K. (before)'.

Japanese tense-aspect forms can be combined with other suffixes e.g. for voice, politeness and negation, as in (9 a–d).

28 The examples in (8 a–b) are based on Nitta et al. (2007).

(9) a. *Tabe-mashi-ta*
eat-POL-PAST
'(I) ate'

b. *Tabe-rare-ta*
eat-PASS-PAST
'(It) got eaten'

c. *Tabe-naka-tta*
eat-NEG-PAST
'(I) didn't eat'

d. *Tabe-rare-mas-en* *deshi-ta*
eat-PASS-POL-NEG COP-PAST
'(It) didn't get eaten'

## 5.2 *Material and method*

The material used was the parallel corpus Parallel Universal Dependencies (PUD) version 2.12 (Zeman et al. 2017), which consists of 1000 sentences gathered from Wikipedia texts and newspaper articles. 750 of these sentences have English as source language. The remaining 250 sentences were gathered from German, French, Italian and Spanish texts, but these have in turn been translated to the other languages via English. The material has been translated to 17 languages, among them Swedish and Japanese, and the UD annotation has been carried out – or has at least been reviewed – manually.

Since we are interested in both exact correspondences and translation divergences, we decided to perform four separate parallel searches using simple queries. First, we used the query

```
TREE_ (FEATS_ "VerbForm=Part") [AND [LEMMA "have", FEATS_
    "Tense=Pres"]]
```

on English-Swedish data to look for occurrences of the English PresPf and their Swedish equivalents.[29] A similar query, capturing the Swedish PresPf construction, was then run on both Swedish-English and Swedish-Japanese data:

```
TREE_ (FEATS_ "VerbForm=Sup") [AND [LEMMA "ha", FEATS_
    "Tense=Pres"]]
```

Searching for the Japanese form *-teiru* proved harder than expected. First of all, PUD tokenizes *-teiru* into *-te* + *-iru* (cf. Section 5.1 for the rationale behind this). Furthermore, the two tokens have several equivalent spellings,

29 This query was already discussed in Section 4.3.1.

which made it impractical to use FORM-based queries. Last but not least, *-teiru* can be conjugated in *-ta* – a form outside the scope of this study – or appear in several negated forms, which are, on the contrary, all relevant for our purposes. Therefore, the *-teiru* query underwent several revisions:

1. Our first attempt was the following:

```
SEQUENCE [XPOS "助詞-接続助詞", XPOS
   "動詞-非自立可能-上一段-ア行"]
```

   This query is based on language-specific POS tags as these are rather fine-grained when it comes to Japanese: *-te* is tagged as 助詞-接続助詞 (*joshi-setsuzokushi*, meaning 'particle-conjunction'),[30] whereas the XPOS tag for *-iru* is 動詞-非自立可能-上一段-ア行 (*doushi-hijiritsukanou jouichidan-agyou*, 'verb – non-independent jouichidangyou paradigm').[31] The `SEQUENCE` keyword combines these two forms and was expected to give us instances of *-teiru*. However, it turned out that results also included instances of the form *-teita* (*-teiru* conjugated in *-ta*, see above). Manual checking also revealed that this query failed to retrieve all instances of *-teiru*: whenever multiple instances of *-teiru* were present in the same sentence, only one was retrieved. This is because `SEQUENCE` queries return the smallest subtree covering the entire given sequence, which often "includes" several otherwise easily distinguishable matches. Negated *-teiru* forms, such as *-teinai*, were also excluded, as the negations imply changes in the second token, *-iru*. For this reason, we decided to base our second attempt on information from the UniDic morphological lexicon (Den et al. 2008), which the treebank stores in the MISC field under the key `UniDicInfo`.

2. This yielded a second, more compact query, which was ultimately the one we used:

```
TREE (MISC "UniDicInfo" "*ている") []
```

   ている is read as *teiru* and refers to the *-teiru* form.[32] This query correctly identified all instances of *-teiru*, including negated forms and

30 Alone, `XPOS "助詞-接続助詞"` also retrieves other forms which work as conjunctions, such as the particle *ga* and the verb form *nagara*.

31 動詞-非自立可能-上一段-ア行 actually refers to verbs with a certain inflectional paradigm and not just the auxiliary *iru*.

32 Here, using a `TREE` query with an empty list of dependents ensures that the token matching the subpattern `MISC "UniDicInfo" "*ている"` is a leaf. This helps better isolating the relevant sentence segments and not getting duplicate hits, since otherwise the whole subtree rooted in *-te* – potentially very extensive – would also be a match (`UniDicInfo` has the same value for both *-te* and *-iru*).

multiple occurrences in the same sentence. However, results also included *-teita-* forms.

3. We tried getting rid of these with a the third query, combining UniDic and XPOS information:

   ```
   SEQUENCE [TREE (MISC "UniDicInfo" "*ている") [], (NOT (XPOS
       "助動詞-助動詞-タ"))]
   ```

   助動詞-助動詞-タ is read as *jodoushi-jodoushi-ta* 'auxiliary-auxiliary-ta' and refers to the -ta form. This query failed however once again in finding negated *-teiru* forms and multiple occurrences in the same sentence. For this reason, we settled upon the second query and manually filtered out *-teita* forms.

## 5.3 *Results*

We present our results pairwise, based on the searches conducted. We start out with the search on English, yielding Swedish correspondences, and move next to Swedish, yielding English correspondences. We then do the same for Swedish and Japanese, i.e. search on Swedish to Japanese, and finally Japanese to Swedish.

We will only focus on *tense-aspect forms* in the Japanese categorization, and disregard other suffixes, e.g. for politeness, voice and polarity as these are not relevant for our purposes. Thus, a verb form such as *tabe-rare-tei-masen deshita* 'It had not been eaten (polite)' will be categorized as *teiru*-form. In the Japanese glossing, we will use the labels -TA and -TEIRU instead of PAST and ASP-NONPAST in order to highlight places where the *-ta* and *-teiru* forms appear (cf. examples 6 b and 6 c). Also, note that Japanese has a different writing system than Swedish and English. To make the results accessible to a broader audience, Japanese examples are romanized.

### 5.3.1 *Swedish equivalents to the English present perfect*

The search on the English PresPf gave 87 hits. The tense forms in the Swedish translation are given in Table 3. *Presens perfekt* occurred in the majority of cases (e.g. *har föreslagit* 'has proposed'), but in six cases *supinum utan hjälpverb* 'supine without auxiliary' appeared instead (e.g. *tidslinjer som flutit runt i medierna*, where *flutit* is the supine form). In one case *preteritum* (*visade* 'showed') appeared and in three cases *enkelt presens* (e.g. the main verb *har* 'have', where English used the present perfect (*e.g. have made*).

Table 3: Swedish equivalents to English present perfect.

| Form | Frequency |
|---|---|
| Presens perfekt | 77 (88,5%) |
| Supinum utan hjälpverb | 6 (6,9%) |
| Enkelt presens | 3 (3,4%) |
| Preteritum | 1 (1,1%) |
| Sum | 87 (100%) |

Table 4: English equivalents to Swedish presens perfekt.

| Form | Frequency |
|---|---|
| Present perfect non-progressive | 75 (94,9%) |
| Present perfect progressive | 3 (3,8%) |
| Simple present | 1 (1,3%) |
| Sum | 79 (100%) |

### 5.3.2 *English equivalents to Swedish presens perfekt*

The search on the Swedish *presens perfekt* gave 79 hits. The tense-aspect forms in the English text that corresponded to the Swedish *presens perfekt* are given in Table 4. 'Non-progressive' in the table refers to cases where the progressive form was not used. The present perfect non-progressive (e.g. *have suggested*) occurred in most cases, but the present perfect progressive (e.g. *has been rising*) also occurred three times. No progressive constructions were used in Swedish in these three cases, but simply *presens perfekt* appeared, e.g. *har ökat* (which can be interpreted both as 'has been rising/has risen'). The only occurrence of the simple present in English was in a passive construction (*the Emperors are often referred to*), where Swedish used the PresPf in a passive construction (*har kejsarna ofta omtalats med*). No cases of the preterite or the pluperfect were found.

### 5.3.3 *Japanese equivalents to Swedish presens perfekt*

The query on the Swedish *presens perfekt* gave 79 hits. The tense-aspect forms in the Japanese text that corresponded the Swedish *presens perfekt* are given in Table 5. As the table shows, a great variety of tense-aspect forms appeared in the Japanese data. The most common form was *-ta* (e.g. *yosoku shi-**ta***

Table 5: Japanese equivalents to the Swedish present perfect.

| Form | Frequency |
|---|---|
| *-ru* | 6 (7,5%) |
| *-ta* | 27 (34,1%) |
| *-teiru* | 24 (30,3%) |
| *-teita* | 3 (3,7%) |
| *-ta koto ga aru* | 1 (1,2%) |
| *-te* | 1 (1,2%) |
| *-te kita* | 3 (3,7%) |
| Verb stem | 7 (8,8%) |
| *-teori* | 3 (3,7%) |
| Other | 4 (5%) |
| Sum | 79 (100%) |

'foresaw'), closely followed by *-teiru* (e.g. *yousei shi-**teiru*** 'have requested').[33] *Presens perfekt* corresponded to *-ta* when the event was set in the past, usually with some sort of result as in (10) below.

(10) a. Swedish
*Röstning **har**, i terrorns jargong, **blivit** det nya mjuka målet*
voting has in the.terror's jargon become the new soft target

b. Japanese
*Touhyou wa, tero no kanten kara suru to, atarashii sofuto taagetto to nat-**ta**.*
voting TOP terror GEN perspective from do if new soft target PART become-TA
'Voting has, in the vernacular of terror, become the new soft target'

In (10), *har blivit* and *nat-ta* express that as a result of 'voting becoming the new soft target', it 'is' the new soft target at present. *Presens perfekt* corresponded to *-teiru* mainly in two cases: (1) with completed past events,

33 Note that the tense-aspect forms of the English translations do not fully mirror the meaning of the Japanese tense-aspect forms entirely, as these are rough translations. For instance, *yousei shi-teiru* in Japanese could be translated as 'requested/have requested/had requested/are requesting' depending on the context. We chose those forms in the English translation which we considered most natural in the case in question.

often with a result at present, as in (11), (2) an event that holds from the past up to present time, as in (12) below.

(11) a. Swedish
*Mer än 5,7 miljoner floridabor* ***har*** *redan*
more than 5,7 million Florida.citizens have already
***gått*** *och* ***röstat*** *efter cirka två veckor med personligt*
gone and voted after about two weeks with personal
*förtidsröstande*
early.voting

b. Japanese
*Yaku 2 shuukan no kijitsumaetouhyou go ni 570*
about 2 weeks GEN early.voting after PART 570
*man ijou no Florida juumin ga sude.ni*
ten.thousand more GEN Florida citizen GEN already
*touhyou shi-****teiru*** *koto ga wakatta*
vote do-TEIRU thing NOM understood
'More than 5.7 million Floridians have already hit the polls after about two weeks of in-person early voting.'

(12) a. Swedish
*Mycket av debatten, från Demokraternas sida i år,*
much of the.debate from the.Democrats' side this year
***har handlat*** *om vit manlig identitet*
has been about white male identity

b. Japanese
*Kotoshi, minshutou gawa kara wa, hakujin dansei*
this.year Democrats side from TOP white.people male
*no aidentiti ni.tsuite ooku no giron ga nas-are-****teiru***
GEN identity about much GEN debate NOM do-PASS-TEIRU
'Much of the debate, from the Democratic side this year, has been about white male identity.'

In (11) *har gått och röstat* and *touhyou shi-teiru* express that the event of 'voting' has already been completed for many participants. In (12), *har handlat om* and *giron ga nasare-teiru* express that the event of 'debating white male identity' has been occurring up to now.

The *-ru*-form appeared six times. An example of this is *tsune ni seijitekina sokumen ga a-****ru*** 'always has had a political dimension', corresponding to Swedish ***har*** [...] *alltid* ***haft*** *en politisk dimension* 'has always had a political dimension'. *-teita* occurred three times, as in where Japanese *rokuten tot-****teita***

Table 6: Swedish correspondences to Japanese *-teiru*.

| Form | Frequency | |
|---|---|---|
| Enkelt presens | 65 | (65%) |
| Preteritum | 11 | (11%) |
| Presens perfekt | 11 | (11%) |
| Skulle + infinitival verb | 2 | (2%) |
| Kommer + infinitival verb | 1 | (1%) |
| Infinitiv verbfras | 1 | (1%) |
| Perfekt infinitiv | 1 | (1%) |
| Other | 8 | (8%) |
| Sum | 100 | (100%) |

'(someone) took six points' corresponded to Swedish *vi* ***har tappat*** *sex poäng* 'we have lost six points'.

There were also cases where *presens perfekt* corresponded to the forms *verb stem* (e.g. *rikai shi* - 'having understood'), *-teori* (e.g. *zouka shi-te ori* 'having increased') and *-te* (*at-te* 'having met').[34]

The only occurrence of *-ta koto ga aru* (*atta koto mo nai* 'there has not been') was conjugated for negation and the *ga*-particle was replaced with *mo* 'also'. The construction *-te kita* occurred three times, as in *chousa shi****-te kita*** 'has investigated up to now', which corresponded to Swedish *har undersökt* 'has investigated'.

The Other category consists of cases where *presens perfekt* did not correspond to a verb, e.g. Japanese *aru Sueeden no kenkyuu ni yoru to* 'according to Swedish research, ...' corresponded to Swedish *en svensk studie har visat att [...]* 'a Swedish study has shown that...'.

### 5.3.4 *Swedish equivalents to Japanese* -teiru

The search for Japanese *-teiru* to Swedish gave 356 hits. Some of the hits were excluded, since the query also included *-teita* forms which are not in the scope of this study. Due to time restrictions we extracted only the first 100 occurrences of *-teiru* that appeared in the concordance. Table 6 shows the Swedish forms that corresponded to Japanese *-teiru*.

The most frequent form in Swedish was *enkelt presens*, which made up the majority of the occurrences. In many of these cases the Swedish *säger* 'says'

34 These are not tense-aspect forms per se, as they do not carry tense-aspect information - the exception being *-te ori*, which carries information for progressive aspect - but since they occurred multiple times we included them in our results.

corresponded to the Japanese verb *nobe-teiru* '(someone) mentions'. *Preteritum* and *presens perfekt* occurred to the same extent. *Preteritum* appeared for instance as *stödde* 'supported', corresponding to Japanese *shiji shi-teiru* 'have supported'. and *presens perfekt* appeared as for instance *har bett* 'has requested', corresponding to Japanese *yousei shi-teiru* 'has requested'.

*Skulle* 'would' + infintival verb appeared twice. In both cases, the auxiliary *ska* was conjugated in the past tense *skulle*, which is why we indicate these occurrences as *skulle* and not *ska*.

*Kommer* 'is going to' + infintival verb occurred only once, which was the occurrence *kommer öka* 'is going to increase', where the auxiliary was conjugated in present tense *kommer* 'come' and not *kom* 'came'. *Infinitiv verbfras* 'infinitive verb phrase' appeared once, where Swedish (*för*) ***att prata*** (*om...*) '(to) talk (about)' corresponded to Japanese [...] *ni tsuite hanashite-ki-**teiru*** 'has talked about'. *Perfekt infinitiv* 'perfect infinitive' also appeared once, where Swedish *att **ha varit** inblandade* 'to have been involved' corresponded to Japanese *juuji shi-**teiru*** 'have been involved in'.

The Other category consists of cases where the *-teiru* form either did not correspond to a verb or it did not correspond to anything at all in the Swedish text. For instance, Swedish *enligt ...* 'according to...' corresponded to Japanese *to houjirare-teiru* 'was reported that...', which is not a verb in Swedish. An example of no correspondence is *kai-teiru* 'writes' in (13) below – which does not have any correspondence in the Swedish text.

(13) a. Japanese
*"Saishin no ibento" wa Reddit, Twitter, Spotify ya*
recent GEN event TOP Reddit Twitter Spotify PART
*sono.ta ooku no ninki websaito ga kanketsu.ni*
other many GEN popular website NOM briefly
*kai-**teiru**, 10 gatsu 21 nichi no tero jiken dearu*
write-TEIRU October 21st GEN terror incident COP
*kanousei ga takai*
probability NOM high

b. Swedish
*De "nyliga händelserna" är troligen attackerna den 21*
the recent events are likely the.attacks the 21
*oktober som tillfälligt slog ut populära*
October which temporarily knocked out popular
*webbsidor som Reddit, Twitter och Spotify såväl som*
websites like Reddit Twitter and Spotify as.well as
*många andra*
many others
'The "recent events" are likely to be the attacks of 21 October that briefly took down popular websites such as Reddit, Twitter and Spotify as well as many others.'

No instances of other tenses, such as *pluskvamperfekt* 'pluperfect' were found.

## 5.4 *Discussion*

In this section, we briefly discuss what tendencies in the use of tense-aspect forms could be seen in our data. Due to space limitations, we restrict our observations to general patterns.

The case study showed similarities and differences in distribution between Swedish and English, and considerable variation between Swedish and Japanese. We refrain from making any in-depth observations on the semantics of tense-aspect forms, but our ambition is to address semantic issues in future studies.

### 5.4.1 *Swedish and English*

Our results showed a strong correspondence between the Swedish and English PresPf. However, the English present perfect was used to a greater extent than the Swedish *presens perfekt*.

The difference seems to depend on both language-specific conventions and translation-induced variation. For instance, the English present perfect in [...] *I* ***have made*** *a full recovery* was translated as *enkelt presens*: [...] ***är*** *jag fullt återhämtad* in Swedish. This ought to be due to the fact that the expression *make a full recovery* is idiomatic and does not have an immediate counterpart in Swedish. Therefore, the translator might have preferred the expression *vara fullt återhämtad* which in the present context sounds more natural in the *enkelt presens* than in *presens perfekt*. Another example is where the English present perfect *But a scan* ***has shown*** *the tumour in Ms Pugh's right lung is growing* [...] was translated to *preteritum* in Swedish: *Men en*

*undersökning* ***visade*** *att tumören i Mr Pughs högra lunga växer* [...]. It is unclear why *preteritum* is used here, but one explanation might be that *preteritum* is preferred in Swedish in narrative texts (Holm 2010).

Furthermore, our results might be due to genre-specific differences. A pilot study we carried out on the LinES treebank, which consists to a great extent of fiction texts, showed that the Swedish *presens perfekt* was used more frequently than the English present perfect. In the future, it would therefore be advantageous to use bigger datasets encompassing various genres to investigate the distribution of tense-aspect forms in the two languages.

### 5.4.2 *Swedish and Japanese*

The Swedish and Japanese results gave a diverse picture of the tense-aspect systems of the two languages. Judging from our results, the Swedish *presens perfekt* mostly overlaps in use with Japanese *-ta* and *-teiru*. In our data, both *-ta* and *-teiru* tended to refer to past events, often with a result holding after the occurrence of the event, and it was in these cases that these forms corresponded to *presens perfekt* (examples 10 and 11). However, one use that *presens perfekt* had in common with *-teiru* but not with *-ta* was to refer to continuative events starting in the past and persisting up to present time (example 12).

Based on our data, Japanese *-teiru* mostly overlaps in use with Swedish *enkelt presens*, chiefly when the two forms refer to an event holding at present. *-teiru* also overlapped in use with *preteritum* and *presens perfekt* when referring to past events, often with a result holding after the occurrence of the event. Since both *presens perfekt* and *preteritum* can often be used in similar contexts (Teleman et al. 1999), the occurrence of *preteritum* was not entirely unexpected. However, it was surprising that *preteritum* and *presens perfekt* occurred to the same extent, since we expected *-teiru* to correspond mainly to Swedish tenses with a present time reference, i.e. *presens* and *presens perfekt*. A possible reason for the unexpected low frequency of *presens perfekt* might be that *-ta* also overlaps in use with *presens perfekt* (as shown in 5.3.3, cf. also Tóth 2020) when referring to past events with a connection to the present, and *-ta* might correspond to *presens perfekt* more frequently than *-teiru* in the corpus. A future investigation on the Swedish correspondences with *-ta* would likely shed light on whether this is the case and if this might depend on semantics, genre, translation issues or other factors.

The unexpected high frequency of *preteritum* might, on the other hand, be due to restrictions on how tense is used in a greater context in Swedish and Japanese. There is a tendency in Swedish to either use so-called presentical tenses, i.e. *presens* and *presens perfekt* or preterital tenses, i.e. *preteritum* and

*pluskvamperfekt* throughout a sequence (Ekerot 2011). In case a text sequence mainly refers to past events, this could facilitate the use of *preteritum* throughout the text, even in cases where *presens perfekt* could be used. Japanese, on the other hand, does not have such a tendency, as past and non-past forms can freely be used throughout a text. Since we do not have access to contextual information in the PUD corpus we unfortunately cannot check whether the use of *preteritum* contra *presens perfekt* is due to textual factors.

By comparing Swedish, English and Japanese we could demonstrate how STUnD can be used to investigate the distribution of tense-aspect forms in languages with fairly similar tense-aspect systems (i.e. Swedish and English) and languages with fairly different tense-aspect systems (i.e. Swedish and Japanese). It needs to be kept in mind that both the Swedish and Japanese texts are translations from English, which likely influences the tense-aspect forms used. Swedish has a morphosyntax similar to that of English, while Japanese morphosyntax differs to a greater extent, which might influence the use of tense-aspect forms depending on how the clauses in the sentence are structured. Even so, we judged that the texts reflected natural language use and the use of tense-aspect in our material should reflect the general use of tense-aspect forms in all three languages.

## 5.5 *Considerations on using STUnD*

The tool was useful for finding corresponding tense-aspect forms between the three languages. However, we encountered issues when Japanese was included in our search.

One strength turned out to be that we could easily identify all instances of PresPf in Swedish and English, since the tool can find dependencies between the auxiliary and supine form in Swedish or the perfect participle in English. Through `TREE` queries, the tool could do this even in cases where several words appeared between the auxiliary and the main verb, as in *har kejsarna ofta omtalats* (approximately 'have the.emperors often been.referred'). The English query allowed us to find occurrences in Swedish where the supine form appeared without an auxiliary. As Figure 4 shows, the tool marks the matching segments and – when possible – their counterpart in the other treebank. This made the search results easily comprehensible. Thus, based on our study, the alignment between Swedish and English, as well as form-based TREE-queries work well in the tool.

While searching for the PresPf in Swedish and English proved easy and gave us the information we wished to retrieve, as we showed in 5.2, the search for *-teiru* in Japanese turned out to be more complicated. As mentioned in the same section, we had to revise the query for Japanese several times in

order to get favourable search results. Some difficulties in this sense, such as homographs making it laborious to look for specific forms through FORM-based queries, are intrinsic to the language itself. Some others, however, are related to the exceptionality of the UD annotation guidelines for Japanese, especially when it comes to tokenization (Guillaume et al. 2024).

One point that pertains to all three languages is that the highlighting was not always precise: the tool sometimes highlighted whole segments in bold and not only the corresponding forms, which still required manual checking. This is due to the very nature of the queries, which always return (unpruned) subtrees as results, and can be mitigated through the application of replacement patterns, which were however challenging to formulate and therefore not used extensively for Japanese. Through the replacement pattern

```
COMPOSE [FILTER_SUBTREES (UPOS "VERB") (UPOS "AUX"), PRUNE TRUE 1]
```

the tool highlighted the Swedish *presens perfekt* and English present perfect forms correctly in most cases. However, in 8 out of 79 cases from Swedish to English and 9 out of 87 cases in the opposite direction, the tool did not highlight any segments in the right-hand-side treebank, even when we used the aforementioned replacement pattern. The same replacement pattern was less successful for Japanese. One-lemma forms such as *-ta* and *-ru* were highlighted correctly, while in the case of forms that UD analysis decomposes as two separate tokens, such as *-teiru*, the tool also highlighted tokens other than the verb form corresponding to the Swedish *presens perfekt*.

In this study, we did not use divergence patterns. In the future, however, these could be useful in searching for translation divergences found in other corpora, such as LinES (Ahrenberg 2007, 2015).

## 6 *Other use cases*

In Section 5, we have shown how STUnD can be used in the context of a cross-lingual study. As mentioned earlier, however, the tool can be used in a variety of contexts, including cases where the parallel corpus of interest is not multilingual. With the hope to provide inspiration for new STUnD users, we present two such applications: grammatical error retrieval and qualitative parser evaluation.

### 6.1 *Grammatical error retrieval*

UD is gaining popularity as an annotation standard for learner corpora, with treebanks already available for second-language (L2) Chinese (Lee, Leung, et al. 2017), English (Berzak et al. 2016, Kyle et al. 2022), Italian (Di Nuovo et al.

2022), Korean (Sung & Shin 2024) and Russian (Rozovskaya 2024) and one under development for Swedish. The main advantage of UD in this setting is that it allows studying the differences and similarities between a learner's first and second language, between different L2s or between standard and learner language. Moreover, the richness of UD annotation is usually sufficient for grammatical error retrieval, without the need for any explicit annotation layer in this sense. This is especially true for *L1-L2 treebanks*, a type of parallel UD corpus where (L2) learner productions are contrasted with expert corrections, assumed to be native-like (L1).

In Section 4.4.2 (cf. Figure 12 in particular), we showed how to use STUnD's diff mode to identify errors in one such corpus, the ESL treebank (Berzak et al. 2016). To complete the picture of how STUnD can be put into use in learner corpus research, this section will demonstrate how parallel queries can help retrieve instances of particular error patterns. Since the ESL treebank lacks morphological tagging, however, we will use VALICO-UD (Di Nuovo et al. 2022), a parallel treebank of L2 Italian.

First of all, many grammatical errors are easily retrieved via simple queries, without needing to compare learner productions with their corrected versions. Agreement errors are a good example of this. The query

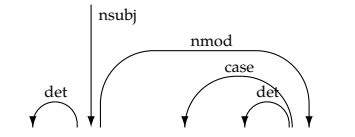
```
TREE_
    (FEATS_ "Gender=Fem")
    [AND [OR [DEPREL_ "det", DEPREL_ "amod"], FEATS_
        "Gender=Masc"]]
```

for instance, matches all subtrees where a feminine token (typically a noun) is introduced by a masculine determiner or modified by the masculine form of an adjective (see Figure 13 for an example of application).

In this case, results are no different from what could be obtained by using any other treebank search tool mentioned in Section 3. Querying parallel L1-L2 treebanks, however, allows users to write simpler, more general queries, whilst limiting search results to grammatical errors. If we were to simplify the query to

```
TREE_ (FEATS_ "Gender=Fem") [FEATS_ "Gender=Masc"]
```

with the goal to look for head-dependent gender agreement errors without specifying the syntactic function of either, the results would include a vast amount of "false positives", such as the sentence fragment

(14)


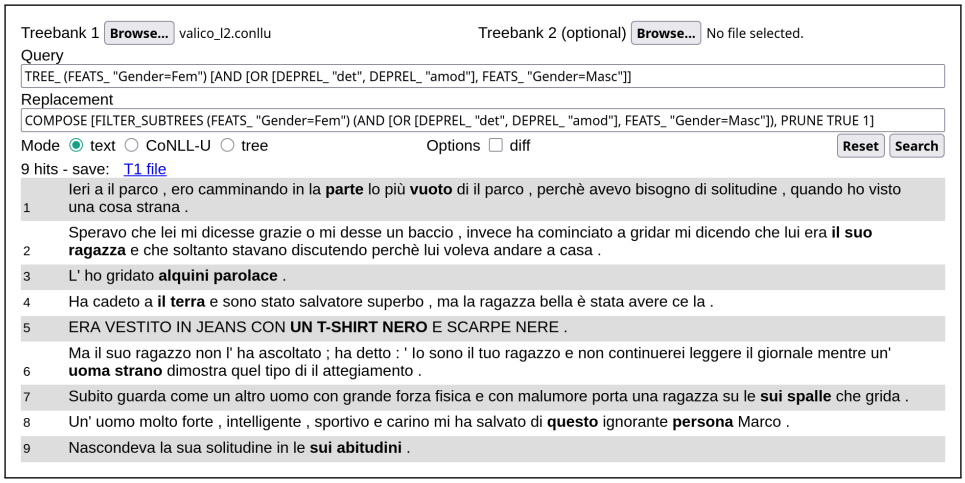



Figure 13: Using STUnD to retrieve agreement errors from L2 productions from the VALICO-UD treebank. In this case, the search is limited to cases where a masculine determiner or adjective is attached to a feminine nominal.

which matches the query not because of an agreement error, but simply because the nominal modifier of the feminine noun *natura* 'nature' is a masculine (*parco*, 'park'). The divergence pattern

```
TREE_ (FEATS_ "Gender=Fem") [FEATS_ "Gender={Fem->Masc}"]
```

on the other hand, limits the results to the cases in which the gender of the dependent is masculine in the original learner sentence but has been corrected to agree with that of its syntactic head in the L1 half of the treebank.

Furthermore, divergence patterns make it possible to identify errors that would be impossible to locate in a monolingual L2 treebanks. The even simpler query

```
FEATS_ "Gender={Fem->Masc}"
```

for example, finds all instances where a feminine token in the L1 half of the treebank corresponds to a masculine one in its L2 counterpart, including cases where the correction involves replacing a correctly inflected but lexically inappropriate masculine noun with a more fitting feminine one, as well as sentences where an entire noun phrase is incorrectly inflected for gender (cf. Figure 14).

## 6.2 *Qualitative parser evaluation and inter-annotator conflict resolution*

So far, we have seen how STUnD can be used on parallel multilingual and L1-L2 treebanks. A third use case is comparing different analyses of the same

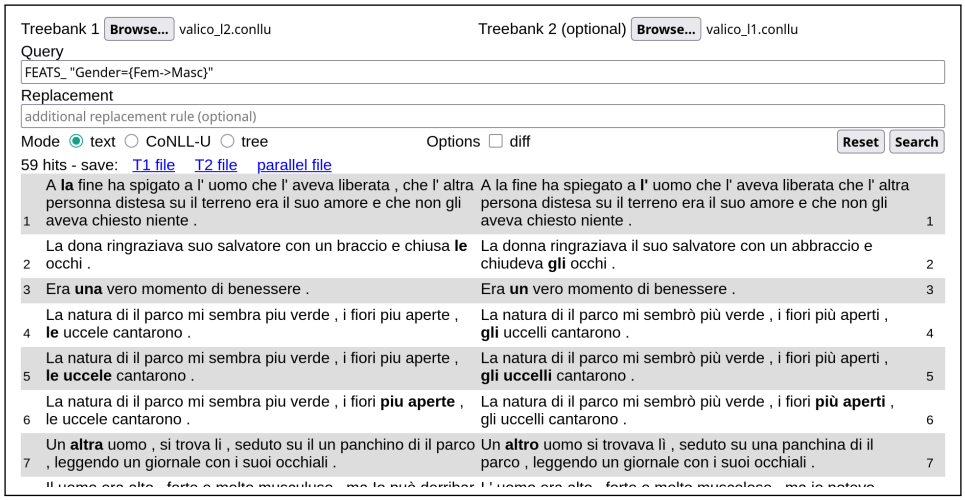



Figure 14: The divergence pattern `FEATS_ "Gender={Fem->Masc}"` run on the VALICO-UD treebank. In many cases, matches are still representative of agreement errors. Hit number 5, however, shows a case in which the entire noun phrase *le uccele* is incorrectly inflected for gender. In the first hit, the two highlighted segments do not correspond to each other, probably due to an annotation error.

surface text. This can be useful both when evaluating automatic parsers on a sentence-by-sentence basis, e.g. by comparing two different models, and when resolving conflicts between human annotators.

In Figures 15–16, an example text in English has been automatically annotated using two different UDPipe2 models.[35] Using STUnD's diff functionality (cf. Section 4.4.2) in text mode gives us a general overview of how similar the analyses are (cf. Figure 15). To gain any insight into how the analyses are different, however, the user needs to switch to either tree or CoNNL-U mode and look beneath the surface. In tree mode, we can see that *Search*, *Tool*, and *Dependencies* are assigned the POS tag `NOUN` on the left side and the tag `PROPN`, i.e. proper noun, on the right (cf. hit number 1 in Figure 16). Some other discrepancies can only be identified by inspecting the CoNNL-U source. The GUM model, for instance, assigns the word *specifically* the feature `Degree=Pos`, which is lacking in the EWT-based analysis. If these were two competing human annotator analyses in the context of a treebanking project, conflicts could be resolved by editing one of the files directly in STUnD.

35 The example text is the abstract of this chapter and the models used are `english-gum-ud-2.15-241121` and `english-ewt-ud-2.15-241121`.

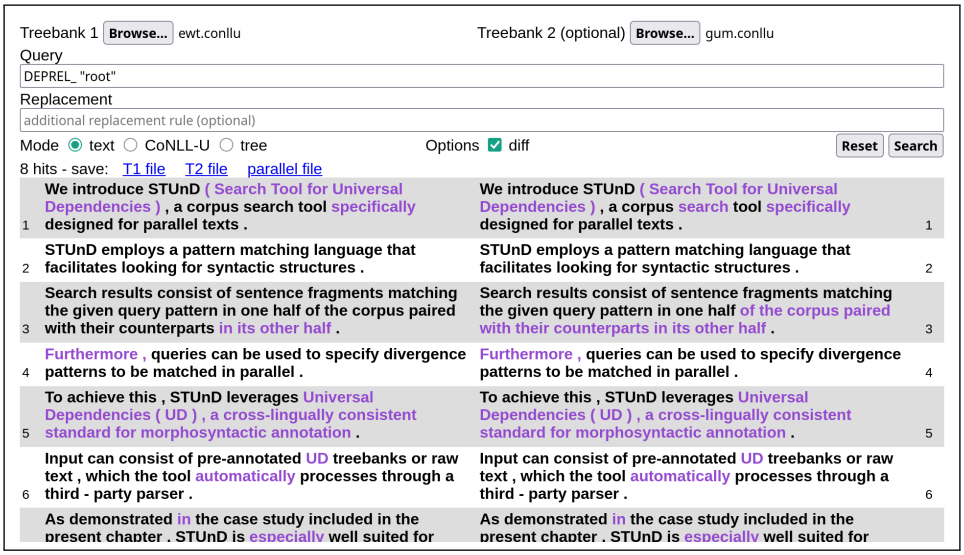



Figure 15: Using diff mode to locate annotation inconsistencies in text mode.

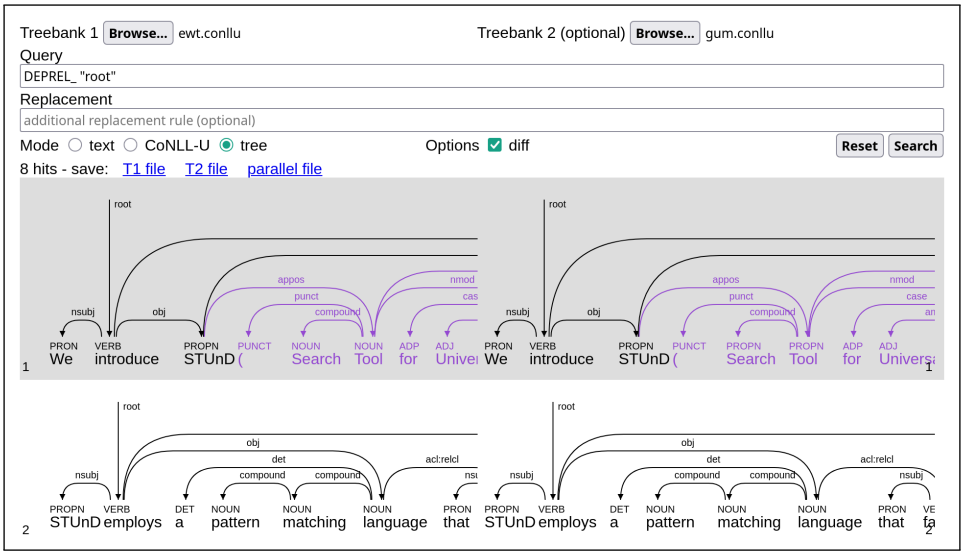



Figure 16: Using diff mode to locate annotation inconsistencies in tree mode.

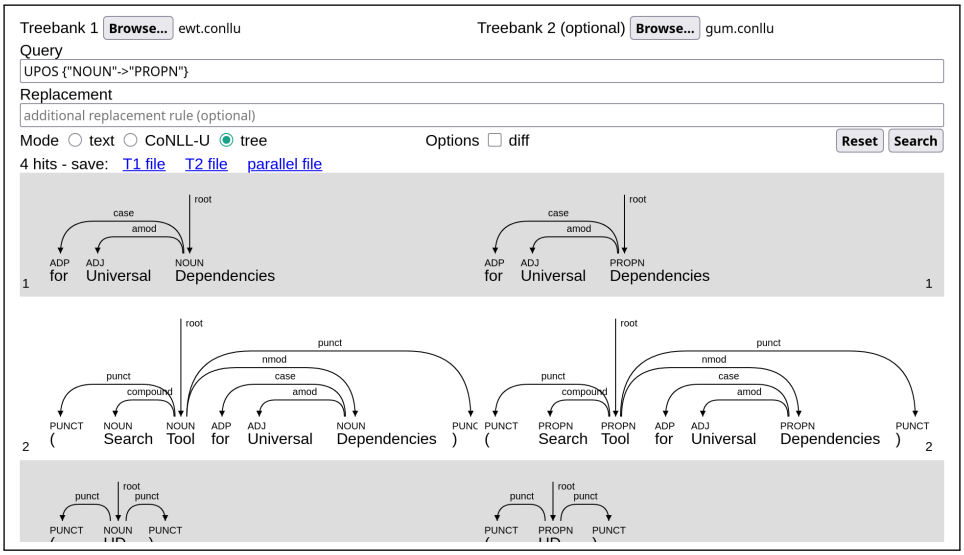



Figure 17: Using divergence patterns to search for specific types of annotation inconsistencies.

Finally, we can use divergence patterns to check whether the differences we have identified are systematic. For example, we can use the following query to look for all cases where a token is classified as `NOUN` by one model and `PROPN` by the other:

```
UPOS "{NOUN->PROPN}"
```

As we can see in Figure 17, there are several occurrences of this phenomenon in this short text, so we can expect this to be due to inconsistencies in the way in which the tags are used in the treebanks on which the two models have been trained.

## 7 *Concluding remarks*

We presented STUnD, a search tool that leverages Universal Dependencies to facilitate the exploration of parallel corpora. At the core of STUnD are two unique features (1) a query language that allows searching for (diverging) syntactic structures in a largely language-agnostic fashion and (2) a subtree alignment algorithm that enables parallel query matching and fine-grained comparisons between two versions of the same text. In addition, STUnD distinguishes itself from most other corpus search tools in that it integrates NLP modules that make it possible to also work on raw text, as well as allow

for the user to interact with the search results by editing them directly in its interface (cf. Sections 4.3.3 and 4.4.3) and exported to a variety of formats, including CoNNL-U itself.

We identified cross-lingual studies as the prime area of application for STUnD. We therefore put the tool to the test in a trilingual case study (cf. Section 5), which shed light on its strengths, but also revealed its current limitations from a user perspective. In particular, the study revealed that query-matching segments extracted from the tool were often larger than one would have hoped and that suitable replacement patterns – which are intended to mitigate this – were challenging to formulate. This, together with the consideration that only a small subset of the available replacement patterns are actually necessary to improve the search results (cf. 4.3.3), prompts us to experiment with automating the refinement step. Queries themselves were sometimes difficult to express, which encourages us to resume experiments already initiated on example-based querying (also referred to as *similar example retrieval*, cf. Masciolini et al. 2023).

In relation to the use of STUnD on learner corpora (Section 6.1), we plan to integrate the experimental (error) pattern extraction functionalities presented in Masciolini et al. (2023) into STUnD's user interface. Finally, STUnD is being tested as a conflict resolution tool in the context of an ongoing treebanking project and will be further developed to meet the annotators' needs. More generally, users and potential contributors are encouraged to follow the development of the tool, report bugs and propose new features.[36]

## *Acknowledgements*

This work is supported and partly funded by Språkbanken – jointly funded by its 10 partner institutions and the Swedish Research Council (2025–2028; project id 2023-00161) as well as (2018–2024; dnr 2017-00626). We are grateful to the Huminfra handbook editors and reviewers for their support throughout the writing process. We would also like to thank our colleagues at Språkbanken Text and the Department for Swedish, Multilingualism, Language Technology at the University of Gothenburg, in particular Niklas Zechner, for their valuable feedback.

36 For an overview of the development process, visit github.com/users/harisont/projects/8. For bug reports and feature request, open an issue at github.com/harisont/STUnD/issues or email arianna@fripost.org.

## *References*

Ahrenberg, Lars. 2007. LinES: An English-Swedish parallel treebank. In Joakim Nivre, Heiki-Jaan Kaalep, Kadri Muischnek & Mare Koit (eds.), *Proceedings of the 16th Nordic Conference of Computational Linguistics* (*NODALIDA 2007*), 270–273. Tartu, Estonia: University of Tartu, Estonia. https://aclanthology.org/W07-2441.

Ahrenberg, Lars. 2015. Converting an English-Swedish parallel treebank to Universal Dependencies. In Joakim Nivre & Eva Hajičová (eds.), *Proceedings of the Third International Conference on Dependency Linguistics* (*Depling 2015*), 10–19. Uppsala, Sweden: Uppsala University. https://aclanthology.org/W15-2103.

Berzak, Yevgeni, Jessica Kenney, Carolyn Spadine, Jing Xian Wang, Lucia Lam, Keiko Sophie Mori, Sebastian Garza & Boris Katz. 2016. Universal Dependencies for learner English. In Katrin Erk & Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (*ACL*) (*Volume 1: Long Papers*), 737–746. Berlin, Germany: Association for Computational Linguistics. DOI: 10.18653/v1/P16-1070.

de Marneffe, Marie-Catherine, Christopher D. Manning, Joakim Nivre & Daniel Zeman. 2021. Universal Dependencies. *Computational Linguistics* 47(2). 255–308. DOI: 10.1162/coli_a_00402.

Den, Yasuharu, Junpei Nakamura, Toshinobu Ogiso & Hideki Ogura. 2008. A proper approach to Japanese morphological analysis: Dictionary, model, and evaluation. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis & Daniel Tapias (eds.), *Proceedings of the Sixth International Conference on Language Resources and Evaluation* (*LREC'08*). Marrakech, Morocco: European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/pdf/258_paper.pdf.

Di Nuovo, Elisa, Manuela Sanguinetti, Alessandro Mazzei, Elisa Corino & Cristina Bosco. 2022. VALICO-UD: Treebanking an Italian learner corpus in Universal Dependencies. *Italian Journal of Computational Linguistics* 8(1). DOI: 10.4000/ijcol.1007. (4 October, 2024).

Ekerot, Lars-Johan. 2011. *Ordföljd, tempus, bestämdhet: Föreläsningar om svenska som andraspråk* [Word order, tempus and definiteness: Lectures on Swedish as a Second Language]. 2nd edition. Malmö: Gleerups.

Guibon, Gaël, Marine Courtin, Kim Gerdes & Bruno Guillaume. 2020. When collaborative treebank curation meets graph grammars. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk & Stelios

Piperidis (eds.), *Proceedings of the Twelfth Language Resources and Evaluation Conference (LREC 2020)*, 5291–5300. Marseille, France: European Language Resources Association. https://aclanthology.org/2020.lrec-1.651/.

Guillaume, Bruno. 2021. Graph matching and graph rewriting: GREW tools for corpus exploration, maintenance and conversion. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL): System Demonstrations*, 168–175. Online: Association for Computational Linguistics. DOI: 10.18653/v1/2021.eacl-demos.21.

Guillaume, Bruno, Kim Gerdes, Kirian Guiller, Sylvain Kahane & Yixuan Li. 2024. Joint annotation of morphology and syntax in dependency treebanks. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti & Nianwen Xue (eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, 9568–9577. Torino, Italia: ELRA & ICCL. https://aclanthology.org/2024.lrec-main.836.

Holm, Lisa. 2010. Presens och preteritum i skönlitterär prosa. In Gunilla Byrman, Anna Gustafsson & Henrik Rahm (eds.), *Svensson och svenskan. Med sinnen känsliga för språk. Festskrift till Jan Svensson den 24 januari 2010*, 49–59. Lund University Publications.

Kolachina, Prasanth & Aarnte Ranta. 2016. From abstract syntax to Universal Dependencies. *Linguistic Issues in Language Technology* 13. DOI: 10.33011/lilt.v13i.1391.

Krsnik, Luka, Kaja Dobrovoljc & Marko Robnik-Šikonja. 2019. *Dependency tree extraction tool STARK 1.0*. Software. http://hdl.handle.net/11356/1284.

Kudou, Mayumi. 1995. *Asupekuto-tensu taikei to tekusuto* [Aspect-tense: System and text]. Tōkyō, Japan: Hitsuji Shobō. DOI: 10.11501/3155604.

Kyle, Kristopher, Masaki Eguchi, Aaron Miller & Theodore Sither. 2022. A dependency treebank of spoken second language English. In Ekaterina Kochmar, Jill Burstein, Andrea Horbach, Ronja Laarmann-Quante, Nitin Madnani, Anaïs Tack, Victoria Yaneva, Zheng Yuan & Torsten Zesch (eds.), *Proceedings of the 17th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2022)*, 39–45. Seattle, Washington: Association for Computational Linguistics. DOI: 10.18653/v1/2022.bea-1.7.

Larsson, Ida & Benjamin Lyngfelt. 2011. Tempus i svenskan [Tense in Swedish]. In *Tid och tidsförhållanden i olika språk* (Studia Interdisciplinaria, Linguistica et Litteraria (SILL) 2), 66–88. Gothenburg, Sweden: Institutionen för Språk och Litteraturer, Göteborgs universitet.

Lee, John, Herman Leung & Keying Li. 2017. Towards Universal Dependencies for learner Chinese. In Marie-Catherine de Marneffe, Joakim Nivre & Sebastian Schuster (eds.), *Proceedings of the NoDaLiDa 2017 Workshop on*

*Universal Dependencies* (*UDW 2017*), 67–71. Gothenburg, Sweden: Association for Computational Linguistics. https://aclanthology.org/W17-0408.

Lee, John, Keying Li & Herman Leung. 2017. L1-L2 parallel dependency treebank as learner corpus. In *Proceedings of the 15th International Conference on Parsing Technologies*, 44–49. Pisa, Italy: Association for Computational Linguistics. https://aclanthology.org/W17-6306.

Lui, Marco & Timothy Baldwin. 2011. Cross-domain feature selection for language identification. In Haifeng Wang & David Yarowsky (eds.), *Proceedings of 5th International Joint Conference on Natural Language Processing*, 553–561. Chiang Mai, Thailand: Asian Federation of Natural Language Processing. https://aclanthology.org/I11-1062.

Lui, Marco & Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In Min Zhang (ed.), *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics* (*ACL*): *System Demonstrations*, 25–30. Jeju Island, Korea: Association for Computational Linguistics. https://aclanthology.org/P12-3005.

Luotolahti, Juhani, Jenna Kanerva, Sampo Pyysalo & Filip Ginter. 2015. SETS: Scalable and Efficient Tree Search in dependency graphs. In Matt Gerber, Catherine Havasi & Finley Lacatusu (eds.), *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics* (*NAACL*): *Demonstrations*, 51–55. Denver, Colorado: Association for Computational Linguistics. DOI: 10.3115/v1/N15-3011.

Masciolini, Arianna. 2023. A query engine for L1-L2 parallel dependency treebanks. In *Proceedings of the 24th Nordic Conference on Computational Linguistics* (*NoDaLiDa*), 574–587. Tórshavn, Faroe Islands: University of Tartu Library. https://aclanthology.org/2023.nodalida-1.57.

Masciolini, Arianna & Aarne Ranta. 2021. Grammar-based concept alignment for domain-specific machine translation. In *Proceedings of the Seventh International Workshop on Controlled Natural Language* (*CNL 2020/21*). Amsterdam, Netherlands: Special Interest Group on Controlled Natural Language. https://aclanthology.org/2021.cnl-1.2.

Masciolini, Arianna & Márton András Tóth. 2024. STUnD: Ett Sökverktyg för Tvåspråkiga Universal Dependencies-trädbanker [STUnD: A search tool for bilingual Universal Dependencies treebanks]. In *Proceedings of the Huminfra Conference* (*HiC 2024*), 95–109. Gothenburg, Sweden: Linköping University Press. DOI: 10.3384/ecp205013.

Masciolini, Arianna, Elena Volodina & Dana Dannélls. 2023. Towards automatically extracting morphosyntactical error patterns from L1-L2 parallel dependency treebanks. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications* (*BEA 2023*), 585–

597. Toronto, Canada: Association for Computational Linguistics. DOI: 10.18653/v1/2023.bea-1.50.

Nitta, Yoshio et al. 2007. *Gendai nihongo bunpō* [Modern Japanese grammar]. Vol. 3: *dai 5-bu. Asupekuto ; dai 6-bu. Tensu ; dai 7-bu. Kōhi* [Part 5: Aspect; Part 6: Tense; Part 7: Polarity]. Tōkyō, Japan: Kuroshio Shuppan.

Pajas, Petr & Jan Štěpánek. 2009. System for querying syntactically annotated corpora. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP 2009): Software Demonstrations*, 33–36. Suntec, Singapore: Association for Computational Linguistics. https://aclanthology.org/P09-4009.

Ranta, Aarne & Prasanth Kolachina. 2017. From Universal Dependencies to abstract syntax. In *Proceedings of the NoDaLiDa 2017 workshop on universal dependencies (UDW 2017)*, 107–116. Gothenburg, Sweden: Association for Computational Linguistics. https://aclanthology.org/W17-0414.

Rozovskaya, Alla. 2024. Universal Dependencies for learner Russian. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti & Nianwen Xue (eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, 17112–17119. Torino, Italia: ELRA & ICCL. https://aclanthology.org/2024.lrec-main.1486.

Shirai, Yasuhiro. 2000. The semantics of the Japanese imperfective -teiru: An integrative approach. *Journal of Pragmatics* 32(3). 327–361. DOI: 10.1016/S0378-2166(99)00051-X.

Sung, Hakyung & Gyu-Ho Shin. 2024. Constructing a dependency treebank for second language learners of Korean. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti & Nianwen Xue (eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, 3747–3758. Torino, Italia: ELRA & ICCL. https://aclanthology.org/2024.lrec-main.332.

de Swart, Henriëtte. 2016. Perfect usage across languages. *Questions and answers in linguistics* 3(2). 57–62.

Teleman, Ulf, Staffan Hellberg & Erik Andersson. 1999. *Svenska akademiens grammatik* (*'Swedish Academy grammar'*), vol. 4 – Satser och meningar. Svenska Akademien. https://svenska.se/grammatik/.

Tóth, Márton András. 2020. *The present perfect in Swedish, Japanese and Hungarian*. Osaka: Osaka university. (MA thesis).

Tóth, Márton András. 2022. Kontinuativt presens perfekt i svenskan [Continuative present perfect in Swedish]. *IDUN – Journal of Nordic Studies* 24. 1–19. DOI: 10.18910/87434. (30 September, 2024).

Zeman, Daniel, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gokirmak, Anna Nedoluzhko, Silvie Cinková, Jan Hajič jr., Jaroslava Hlaváčová, Václava Kettnerová, Zdeňka Urešová, Jenna Kanerva, Stina Ojala, Anna Missilä, Christopher D. Manning, Sebastian Schuster, Siva Reddy, Dima Taji, Nizar Habash, Herman Leung, Marie-Catherine de Marneffe, Manuela Sanguinetti, Maria Simi, Hiroshi Kanayama, Valeria de Paiva, Kira Droganova, Héctor Martínez Alonso, Çağrı Çöltekin, Umut Sulubacak, Hans Uszkoreit, Vivien Macketanz, Aljoscha Burchardt, Kim Harris, Katrin Marheinecke, Georg Rehm, Tolga Kayadelen, Mohammed Attia, Ali Elkahky, Zhuoran Yu, Emily Pitler, Saran Lertpradit, Michael Mandl, Jesse Kirchner, Hector Fernandez Alcalde, Jana Strnadová, Esha Banerjee, Ruli Manurung, Antonio Stella, Atsuko Shimada, Sookyoung Kwak, Gustavo Mendonça, Tatiana Lando, Rattima Nitisaroj & Josie Li. 2017. CoNLL 2017 shared task: Multilingual parsing from raw text to Universal Dependencies. In Jan Hajič & Dan Zeman (eds.), *Proceedings of the CoNLL 2017 shared task: Multilingual parsing from raw text to universal dependencies*, 1–19. Vancouver, Canada: Association for Computational Linguistics. DOI: 10.18653/v1/K17-3001. https://aclanthology.org/K17-3001/.

## *List of abbreviations*

### *Abbreviations used in glosses*

| | |
|---|---|
| ACC | accusative particle |
| ASP | aspectual inflection |
| COP | copula |
| GEN | genitive particle |
| NOM | nominative particle |
| PART | particle with various functions |
| TOP | topicalizing particle |

### *Other*

| | |
|---|---|
| NLP | Natural Language Processing |
| POS | Part of Speech |
| UD | Universal Dependencies |
| PUD | Parallel Universal Dependencies |

| | |
|---|---|
| CoNNL | Conference on Computational Natural Language Learning |
| L1 | first language |
| L2 | second or foreign language |
| TSV | Tab-Separated Values |
| SVG | Scalable Vector Graphics |

## 8 *Corresponding authors*

Arianna Masciolini
Språkbanken Text
Department of Swedish, Multilingualism, Language Technology
University of Gothenburg
arianna.masciolini@gu.se

Herbert Lange
Språkbanken Text
Department of Swedish, Multilingualism, Language Technology
University of Gothenburg
herbert.lange@gu.se

Márton András Tóth
Department of Swedish, Multilingualism, Language Technology
University of Gothenburg
marton.toth@gu.se