Towards automatically extracting morphosyntactical error patterns from L1-L2 parallel dependency treebanks

Arianna Masciolini, Elena Volodina and Dana Dannélls

Språkbanken Text, Department of Swedish, Multilingualism, Language Technology, University of Gothenburg

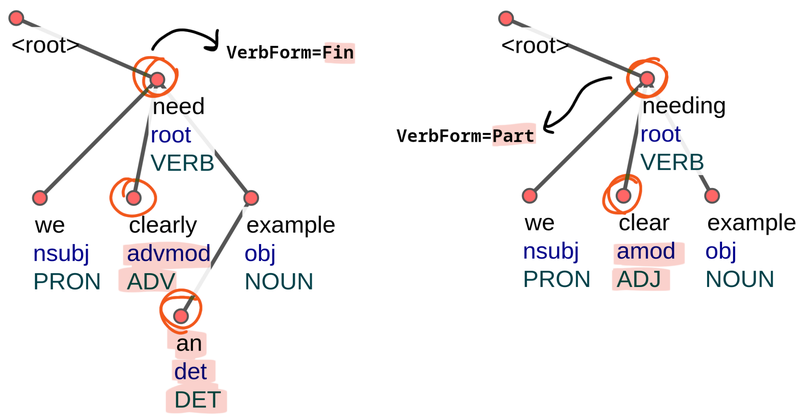
Abstract

L1-L2 parallel dependency treebanks are UD-annotated corpora of learner sentences paired with correction hypotheses. Automatic morphosyntactical annotation has the potential to remove the need for explicit manual error tagging and improve interoperability, but makes it more challenging to locate grammatical errors in the resulting datasets. We therefore propose a novel method for automatically extracting morphosyntactical error patterns and perform a preliminary bilingual evaluation of its first implementation through a similar example retrieval task. The resulting pipeline is also available as a prototype CALL application.

L1-L2 parallel dependency treebanks

- learner sentences paired with correction hypotheses
- no explicit error labelling, just morphosyntactical annotation
- main design goal: interoperability → Universal Dependencies annotation

Example error-correction tree pair:



⟨L1: "we clear*ly* needing *an* example", L2: "we clear needing _ example"⟩

Goal

Extracting **machine-readable error patterns** to be used in explainable GEC, controlled feedback comment generation and more.

Step 1: error detection

1. use the `concept-alignment` package to extract subtree and head alignments
2. select discrepant alignments

For example (discrepancies in bold):

- ⟨we **clearly need an example**, we **clear needing example**⟩*, ⟨**need**, **needing**⟩*
- ⟨we, we⟩
- ⟨**clearly**, **clear**⟩*
- ⟨**an** example, example⟩*, ⟨example, example⟩

Step 2: patten generation

To describe errors, we extend a pre-existing query language for UD treebanks.

The `gf-ud` pattern matching language

pattern type	example
single-token patterns	`DEPREL "nsubj"`
tree patterns	`TREE (POS "NOUN") [DEPREL "det"]`
sequence patterns	`SEQUENCE [DEPREL "advmod", POS "VERB"]`
logical operators	`OR [POS "NOUN", POS "PRON"]`

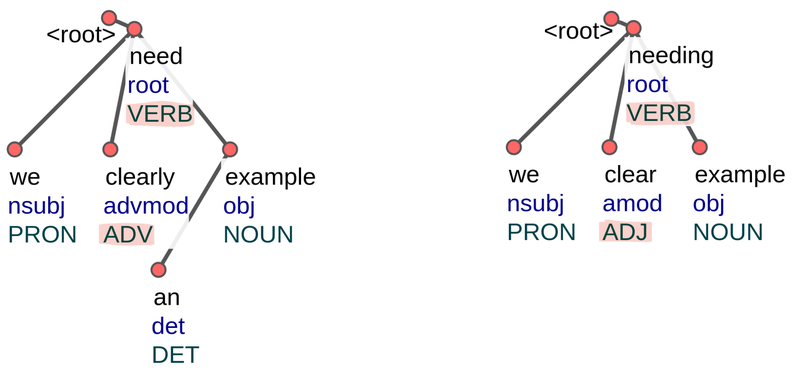
Extension: L1-L2 UD patterns

Errors are represented as pairs of UD patterns, such as:

⟨`TREE_ (POS "VERB") [POS "ADV"]`, `TREE_ (POS "VERB") [POS "ADJ"]`⟩

or `TREE_ (POS "VERB") [POS "{ADV -> ADJ}"]` in short,

meaning that the verb should be modified by an adverb rather than an adjective.



Simplification strategies

Converting subtree pairs to L1-L2 patterns often results in overly specific error descriptions → automatic simplification by:

1. filtering by CoNNL-U field
2. removing CoNNL-U fields whose values are identical everywhere in the L1 and L2 component of the pattern
3. eliminating identical subpatterns
4. simplifying lists of length 1, tree patterns with empty dependent lists etc.

Preliminary evaluation

- evaluation through a **similar example retrieval task**:
  1. **extract** L1-L2 patterns from an error-correction input pair
  2. **query** an L1-L2 treebank with the extracted pattern
- pipeline also available as a prototype CALL application
- simplifying assumptions:
  - only morphosyntactical errors
  - one error per sentence → UD-parsed **linguistic acceptability datasets**:

name	language	size	description
BLiMP	English	14 996	artificially generated sentences
DaLAJ	Swedish	1 198	postprocessed L2 learner sentences

- evaluation metrics:
  - R (retrieval rate): percentage of input pairs with one or more matches
  - R_+ (successful retrieval rate): percentage of input pairs with one or more *correct* matches

Results

	BLiMP	DaLAJ
R	82%	69%
R_+	82%	63%

Error analysis:

- some syntactical errors (word order, missing subject and ADJ→ADV) cause issues at the parsing stage
- nonexistent word forms often handled incorrectly
- better results on English probably due to the dataset (more example sentences, controlled generation, lexical identity between L1 and L2)

Future work

- pattern extraction method:
  - better handling of nonexistent word forms
  - real-world L2 data (overlapping errors, non-grammatical errors...)
- example retrieval application:
  - pattern selection/ranking
  - user interface
- improvement of automatic annotation of L2 sentences
- use of patterns for automatic feedback comment generation

Learn more

full paper

code

gf-ud pattern matching language

concept-alignment package