

Identifying Grammar Rules for Language Education with Dependency Parsing in German

Eleni Metheniti, Pomi Park, Kristina Kolesova, Günter Neumann

August 27, 2019

Depling – SyntaxFest

Why identify grammar rules?

- Our larger mission: find appropriate texts for first language learners in primary school education (in German and other languages...)
- *Text difficulty* relies on many aspects; sentence length, word frequency, syntax...
- How can we determine the difficulty of a sentence given the existing grammar rules (syntax + morphosyntax)?
- How to even find grammar rules in text?

Our approach

- Create a new *query language* for translating grammar rules to syntactic/morphosyntactic patterns
- Build *patterns* for German syntactic phenomena (age appropriate) + assign them with difficulty
- Build a *matching algorithm* where *input*: parses + patterns, *output*: grammar rules
- *Evaluate* the matches with gold parses and parses from 4 parsers (MUNDERLINE, UDPIPE, JPTDP, TURKU)

Our main goals

- Create very restrictive patterns that would not be found erroneously/overzealously
- Descriptive and human-readable patterns
- Search on *dependency parses* (**not** just string methods or regex!)
- Fast and successful with our dependency parser

Why create a new query language?

- **Most** existing text query languages (ANNIS, Poliqarp, COSMAS II) do not support dependency parsing, use regex (too difficult), match meaningless strings...
- **PML-TQ** (Pajas and Štěpánek, 2009): very robust, too complex
- **TüNDRA** (Martens, 2012): almost perfect... but not quite (uses TIGER annotation, allows surface structure)

Query language template

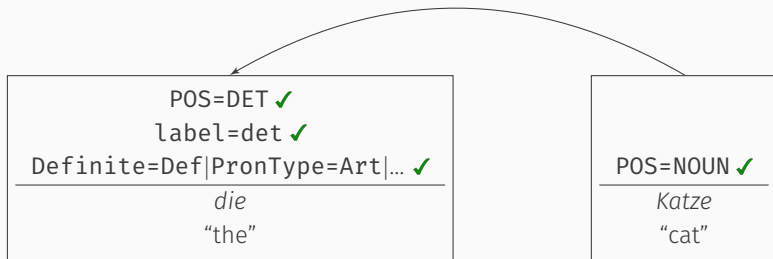
```
comp_word: POS={A,B}&  
            label={c}&  
            feature={d,e}&  
            lemma={'e'}&  
            wordform={'f','g'}&  
            wordform={h-,i-}&  
            wordform={-j-},  
head_word: POS={A,B}&  
            label={c}&  
            feature={d,e}&  
            lemma={'e'}&  
            wordform={'f','g'}&  
            wordform={h-,i-}&  
            wordform={-j-},  
tokenID(head_word) = headID(comp_word)
```

Figure 1: General template for a pattern with a *head-dependent* relation.

Example 1: NP with definite determiner

comp_word: POS={DET}&label={det}&
 feature={Definite=Def,PronType=Art},
head_word: POS={NOUN},
tokenID(head_word) = headID(comp_word)

Figure 2: Pattern to identify a noun phrase with a definite article.

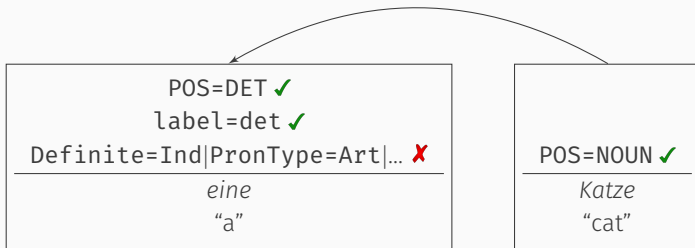


Match!

Example 1: NP with definite determiner (complex pattern)

comp_word: POS={DET}&label={det}&
 feature={Definite=Def,PronType=Art},
head_word: POS={NOUN},
tokenID(head_word) = headID(comp_word)

Figure 3: Pattern to identify a noun phrase with a definite article.



No match.

Example 2: Definite pronoun (simple patterns)

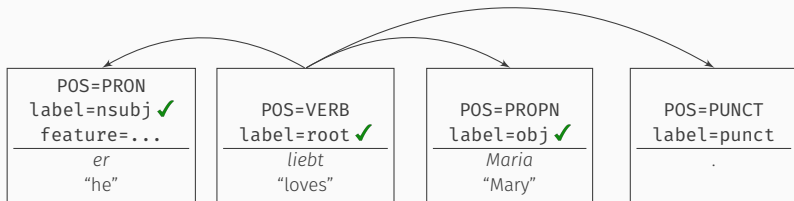
head_word: POS={DET}&label={det}&
feature={Definite=Def,PronType=Art}

Figure 4: Pattern to identify a definite determiner.

POS=DET ✓ label=det ✓ Definite=Def PronType=Art ... ✓
<i>die</i> "the"

Example 3a: Transitive sentence (compound patterns)

comp_word: label={nsubj},
head_word: POS={VERB}&label={root},
tokenID(head_word) = *headID(comp_word)*
AND
comp_word: label={obj},
head_word: POS={VERB}&label={root},
tokenID(head_word) = *headID(comp_word)*



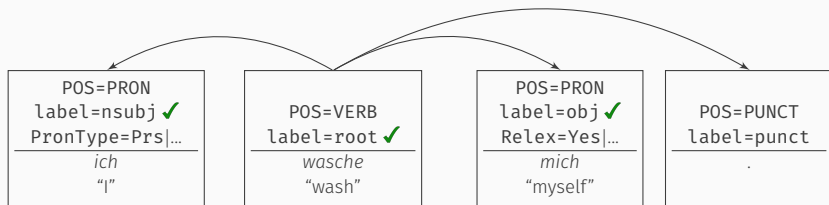
Match!

Example 3b: Reflexive sentence

comp_word: label={nsubj},
head_word: POS={VERB}&label={root},
tokenID(head_word) = *headID(comp_word)*

AND

comp_word: label={obj},
head_word: POS={VERB}&label={root},
tokenID(head_word) = *headID(comp_word)*



Also a match...

Exclude operator

- Distinguish similar patterns by **excluding** the parts of the pattern that should not match
- Example: Pattern to **only** match simple mono-transitive sentences (not bitransitive sentences, not reflexive sentences)

```
comp_word:  label={nsubj},  
head_word:  POS={VERB}&label={root},  
tokenID(head_word) = headID(comp_word)
```

AND

```
comp_word:  label={obj},  
head_word:  POS={VERB}&label={root},  
tokenID(head_word) = headID(comp_word)
```

AND

```
~(comp_word:  label={iobj},  
head_word:  POS={VERB}&label={root},  
tokenID(head_word) = headID(comp_word))
```

AND

```
~(comp_word:  label={obj,iobj}& feature=  
{PronType=Prs,Reflex=Yes},  
head_word:  POS={VERB}&label={root},  
tokenID(head_word) = headID(comp_word))
```

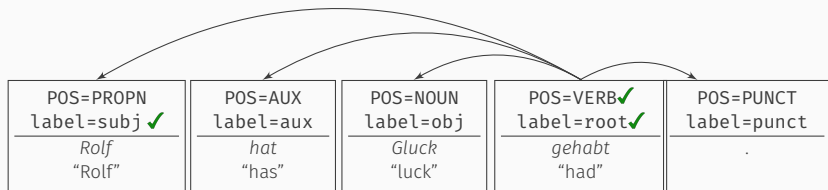
Example 3a, revised

comp_word: label={nsubj},
head_word: POS={VERB}&label={root},
tokenID(head_word) = headID(comp_word)

AND
comp_word: label={obj},
head_word: POS={VERB}&label={root},
tokenID(head_word) = headID(comp_word)
AND

~(*comp_word*: label={iobj},
head_word: POS={VERB}&label={root},
tokenID(head_word) = headID(comp_word))

AND
~(*comp_word*: label={obj,iobj}& feature={PronType=Prs,Reflex=Yes},
head_word: POS={VERB}&label={root},
tokenID(head_word) = headID(comp_word))



Part 1: Match!

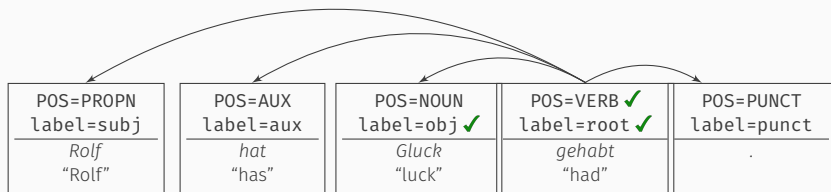
Example 3a, revised

comp_word: label={nsubj},
head_word: POS={VERB}&label={root},
tokenID(head_word) = *headID(comp_word)*

AND
comp_word: label={obj},
head_word: POS={VERB}&label={root},
tokenID(head_word) = *headID(comp_word)*
AND

~(*comp_word*: label={iobj},
head_word: POS={VERB}&label={root},
tokenID(head_word) = *headID(comp_word)*)

AND
~(*comp_word*: label={obj,iobj}& feature=
{PronType=Prs,Reflex=Yes},
head_word: POS={VERB}&label={root},
tokenID(head_word) = *headID(comp_word)*)



Part 2: Match!

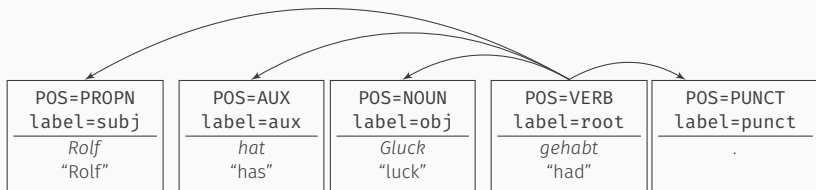
Example 3a, revised

comp_word: label={nsubj},
head_word: POS={VERB}&label={root},
tokenID(head_word) = *headID(comp_word)*
AND

comp_word: label={obj},
head_word: POS={VERB}&label={root},
tokenID(head_word) = *headID(comp_word)*
AND

~(*comp_word*: label={iobj},
head_word: POS={VERB}&label={root},
tokenID(head_word) = *headID(comp_word)*)
AND

~(*comp_word*: label={obj,iobj}& feature=
{PronType=Prs,Reflex=Yes},
head_word: POS={VERB}&label={root},
tokenID(head_word) = *headID(comp_word)*)



Part 3: *comp_word* not found → No match.

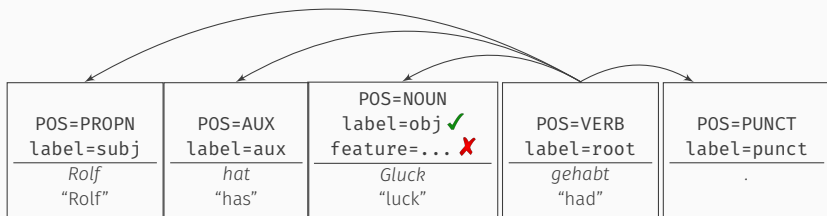
Example 3a, revised

comp_word: label={nsubj},
head_word: POS={VERB}&label={root},
tokenID(head_word) = *headID(comp_word)*

comp_word: label={obj},
head_word: POS={VERB}&label={root},
tokenID(head_word) = *headID(comp_word)*

~(*comp_word*: label={iobj},
head_word: POS={VERB}&label={root},
tokenID(head_word) = *headID(comp_word)*)
AND

~(*comp_word*: label={obj,iobj}& feature=
{PronType=Prs,Reflex=Yes},
head_word: POS={VERB}&label={root},
tokenID(head_word) = *headID(comp_word)*)



Part 4: Partial match → No match.

135 patterns for syntactic/morphosyntactic German grammar rules

ID	Description	Dif.	Pattern
222	Auxiliary verb "haben", present indicative	1	head_word: POS={AUX} & wordform={"hab","habe","hast","hat","haben"} & feature= {Mood=Ind,VerbForm=Fin}
240	Composed forms: Perfect indicative	1	comp_word: {<222>,<218>}, head_word: POS={VERB} & feature={VerbForm=Part}, tokenID(head_word)=headID(comp_word)
289	Simple clause with intransitive verb, with auxiliary verb	1	(comp_word: label={nsubj}, head_word: POS={VERB}&label={root}, tokenID(head_word)=headID(comp_word)) AND (comp_word: POS={AUX} & label={aux}, head_word: POS={VERB} & label={root} & feature={VerbForm=Part}, tokenID(head_word)=headID(comp_word)) AND ~(head_word: label={obj}) AND ~(head_word: label={iobj}) AND ~(head_word: POS={PUNCT}&wordform={"?"}) AND ~(head_word: feature={Mood=Imp}&label={root})

For string matches, lemma matches, affix matches:

- Dictionary of 15K German words from 117K corpus of children's texts
- Every word has orthographic, phonological, morphological information

NB: Word matches **should** be used sparingly; dependencies are favoured.

The case of MWEs

- Gold treebanks annotate **MWEs** with **compound**, but some parsers fail to.
- Also, what about lexicalized phrases?
- We pre-process dependency parse inputs before matching with patterns to **merge** lexicalized phrases.

1. Compose list of relevant phrases + their *head* word
2. Find phrase in dependency parse
3. Concatenate words to one “word” with *head* features

1	As	ADV	_	2	advmod	} 2	As soon as	ADV	Degree=Pos	22	advmod
2	soon	ADV	Degree=Pos	22	advmod						
3	as	SCONJ	_	9	mark						
4	...						4	...			

Implementation

- Implementation of the algorithms in Python3
- Dependency parsing: MUNDERLINE (Volkh and Neumann, 2012)
- Available to the partners through API and online tool

Matcher Demo [About](#)

Sie belasten die Umwelt.

Match

sie belasten die umwelt .

1	Sie	_	PRON	_	Case=Nom Person=2 Poli... 2	nsubj	_	_	O	_	
2	belasten	_	VERB	_	Number=Plur Person=3 V... 0	root	_	_	O	_	212 Personal Pronouns,Nominative
3	die	_	DET	_	Case=Acc Definite=Def G... 4	det	_	_	O	_	265 Personal pronouns as subject
4	Umwelt	_	NOUN	_	Case=Acc Gender=Fem ... 2	obj	_	_	O	_	205 Definite article
5	.	_	PUNCT	_	_	punct	_	_	O	_	269 Noun phrase with definite article

Rules:
290 Simple clause, transitive verb

- How well does the matcher work?
- How well our parser performs compared to others?
 - Evaluation on gold parses
 - Evaluation on parsers:
 - **Munderline** Volokh and Neumann (2012)
 - **UDPipe** (Straka and Straková, 2017)
 - **jPTDP** (Nguyen and Verspoor, 2018)
 - **Turku** neural parser pipeline (Kanerva et al., 2018)

Results on gold parses

Dev set: 152 sentences + Test set: 101 sentences

	<i>Gold</i>
<i>Total</i>	978
<i>TP</i>	922
<i>FP</i>	26
<i>FN</i>	56
<i>Precision</i>	0.9726
<i>Recall</i>	0.9427
<i>F1</i>	0.9574

Table 1: Dev set results

	<i>Gold</i>
<i>Total</i>	776
<i>TP</i>	734
<i>FP</i>	32
<i>FN</i>	42
<i>Precision</i>	0.9582
<i>Recall</i>	0.9459
<i>F1</i>	0.9520

Table 2: Test set results.

Results on parsers

- **Turku** most successful (Precision, Recall)
- **jPTDP** doesn't output morphological features

	<i>Gold</i>	<i>MUNDERLINE</i>	<i>UDPipe</i>	<i>jPTDP</i>	<i>Turku</i>
Total	978	978	911	978	911
TP	922	742	638	249	788
FP	26	92	105	98	89
FN	56	236	273	729	123
Prec.	0.9726	0.8897	0.8587	0.7176	0.8985
Rec.	0.9427	0.7587	0.7003	0.2546	0.8650
F1	0.9574	0.8190	0.7715	0.3758	0.8814

Table 3: Results on dev set.

Results on parsers

- Turku most successful (Precision, Recall)
- jPTDP doesn't output morphological features

	<i>Gold</i>	<i>MUNDERLINE</i>	<i>UDPipe</i>	<i>jPTDP</i>	<i>Turku</i>
<i>Total</i>	776	776	664	776	664
<i>TP</i>	734	601	496	246	587
<i>FP</i>	32	80	89	68	96
<i>FN</i>	42	175	168	530	77
<i>Prec.</i>	0.9582	0.8825	0.8479	0.7834	0.8594
<i>Rec.</i>	0.9459	0.7745	0.7470	0.3170	0.8840
<i>F1</i>	0.9520	0.8250	0.7942	0.4514	0.8716

Table 4: Results on test sets.

Discussion: Why not 100% accuracy?

- The selected rules have limited scope
e.g. Prepositional phrases with ellipsis not found
- Dependency parsing choices:
 - Conjunctions: second part is `conj` (now fixed!)
 - No position information: advantage and disadvantage
- Gold parses vs. Parser output:
 - MWE expressions (addressed)
 - Multiword tokens (“zum” → “zu” + “dem”) (now fixed!)

- Integration to the text difficulty evaluation
- Patterns and resources for Greek, English, Spanish
- Improvement and extension of our patterns, use in other NLP applications...

Thank you to our coworkers, our SyntaxFest reviewers, and **you!**

Questions?

Demo?

1	C'	ce	PRON	_	Number=Sing Person=3 PronType=Dem	4	nsubj	_
2	est	être	AUX	_	Mood=Ind Number=Sing Person=3 Tense=Pres VerbForm=Fin	4	cop	_
3	la	le	DET	_	Definite=Def Gender=Fem Number=Sing PronType=Art	4	det	_
4	fin	fin	NOUN	_	Gender=Fem Number=Sing	0	root	_
5	!	!	PUNCT	_	_	4	punct	_

- Kanerva, J., Ginter, F., Miekka, N., Leino, A., and Salakoski, T. (2018). Turku neural parser pipeline: An end-to-end system for the conll 2018 shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 133–142.
- Martens, S. (2012). TüNDRA: TIGERSearch-style treebank querying as an XQuery-based web service. In *Proceedings of the joint CLARIN-D/DARIAH Workshop 'Serviceoriented Architectures (SOAs) for the Humanities: Solutions and Impacts', Digital Humanities*.
- Nguyen, D. Q. and Verspoor, K. (2018). An Improved Neural Network Model for Joint POS Tagging and Dependency Parsing. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 81–91, Brussels, Belgium. Association for Computational Linguistics.
- Pajas, P. and Štěpánek, J. (2009). System for querying syntactically annotated corpora. In *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, pages 33–36. Association for Computational Linguistics.
- Straka, M. and Straková, J. (2017). Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Volokh, A. and Neumann, G. (2012). Transition-based Dependency Parsing with Efficient Feature Extraction. In *35th German Conference on Artificial Intelligence (KI-2012)*, Saarbrücken, Germany.