Identifying grammar rules for language education with dependency parsing in German

Eleni Metheniti **Pomi Park** **Kristina Kolesova** **Günter Neumann**

DFKI
Stuhlsatzenhausweg 3
66123, Saarbrücken
`firstname.lastname@dfki.de`

Abstract

We propose a method of determining the syntactic difficulty of a sentence, using syntactic patterns that identify grammatical rules on dependency parses. We have constructed a novel query language based on *constraint-based dependency grammars* and a grammar of German rules (relevant to primary school education) with patterns in our language. We annotated these rules with a difficulty score and grammatical prerequisites and built a *matching* algorithm that matches the dependency parse of a sentence in CoNLL-U format with its relevant syntactic patterns. We achieved 96% precision and 95% recall on a manually annotated set of sentences, and our best results on using parses from four parsers are 88% and 84% respectively.

1 Introduction

Language teaching on beginner and elementary levels, even for native speakers, brings the challenge of presenting grammatical phenomena which are familiar, unconsciously familiar or unknown to the learner, in a formal and repetitive way so that the learner will be able to understand and remember them. The presentation of these phenomena to the learner should be consistent, to establish correct patterns, repeated, to facilitate learning, and of gradual difficulty and infrequency, to ensure that the easier structures are acquired before the more difficult ones. The iRead project, in which we are scientific partners, aims to create learning applications for children in primary education, in which the user will be able to read and play with language content tailored to their learning needs, e.g. games that require the user to choose the correct morpheme, phoneme or part-of-speech to complete a pattern. Our roles are, first, to provide learning resources for native German primary school learners (ages 6-9) and, second, to provide a syntactic tool for the analysis of sentences and texts (a CoNLL–U multilingual dependency parser (Volokh and Neumann, 2012)) and a formalism that can be used to represent grammatic phenomena and query them from dependency parses. In this paper, we will be focusing on how we created our syntactic pattern formalism, the algorithm to match patterns with sentences, and the language resources that we used alongside our pattern matching tool, in order to find the grammatical rules that are applicable in a sentence.

The reason we decided to create our own query language was the need to be able to create very restrictive patterns that would almost never be found in the text erroneously or overzealously; these patterns express grammatical phenomena taught in school to young learners, and our margin for incorrect matches of a grammar rule with text is very limited. In addition, our language should be very descriptive but also human readable, so that our partners will be able to create grammatical patterns for other languages without extensive knowledge on logical operators and regular expressions. Finally, we opted to create a query language whose search relies on dependency parsing, and not on the surface structure of a clause. We will present our query language and the grammatical rule patterns that we have created for German primary school learners, and we will also present the matching algorithm we built to match these rule patterns to sentences from our corpus of children's texts. Moreover, we will be evaluating our matching algorithm's performance on this corpus with parses from our and other parsers; the reason we are not using more complex text is because our patterns are made to reflect syntactic phenomena appropriate for child learners.

2 Related Work

We are aware that many query languages have been created over the years, in which the researcher can create a pattern to extract one or multiple words with specific syntactic, morphological, orthographic etc. features from text. However, most of them do not support queries from dependency parses, but require annotated text with parts-of-speech, and only a few such as *ANNIS* (Zeldes et al., 2009) allow for patterns to look for relationships between two nodes of a syntactic tree. Other languages require the position of extra words given explicitly relative to the first word (*COSMAS II*; (Bodmer, 1996)), or rely on neighbouring words without capturing any dependencies (*Poliqarp*; (Przepiórkowski et al., 2004)). In addition, these query languages require a certain level of expertise with regular expressions and the syntax of the language; efforts have been made to simplify the syntax of these languages, for example *Coral* (Kuhn and Höfler, 2012) is a controlled natural language that translates natural language queries to the *ANNIS* syntax.

Query languages tailored for use with dependency parses also have existed for a while; for example *PML-TQ* (Pajas and Štěpánek, 2009) contains a very robust query language which is able to search for one, two or multiple constituents of a syntactic tree, either terminal non-terminal nodes. It is versatile and dynamic, and it would allow us to define patterns between words and phrases to cover the simplest rules (e.g. the presence of predicate) to more complex (e.g. constituents of a question clause), but its syntax is very complex for us to use throughout our project. *TüNDRA* (Martens, 2012) is another query language which also supports queries of one or multiple words based on annotation, deep or surface structure, negation, etc. and uses a similar syntax and the TIGERSearch annotation schema (Lezius, 2002). For our intents and purposes, it would be a fairly complete approach for our task of querying grammatical rules; however, we still wanted to attempt an approach that would be inspired by the successes of the predecessors and offer even better readability and adherence to the theory of dependency parsing, instead of also offering a search for serialized words, syntactically meaningless strings etc.

To create the queries for the grammatical rules, as explained in Section 1, we avoided the use of an automatic method to extract syntactic patterns automatically from text. Pattern induction would not be accurate and informative enough to create patterns for the specific grammatic rules that we have declared. For example, a statistical extraction (Ammar, 2016) that created pairs of a *dependent* and *head* word from dependency parses of English sentences would probably not be sufficient in capturing all the constituents of a grammatical rule, and in any case would require human annotation to the corresponding grammatical rule and its difficulty and frequency. A statistic approach close to our needs involves extracting syntactic patterns based on syntax trees from a large English corpus and scoring their difficulty based on a Zipfian distribution (Kauchak et al., 2017). However, as they discuss in their paper and in previous research (Kauchak et al., 2012), frequency is a solid but not determining factor to the difficulty of a pattern, and surface syntactic structure is not sufficient to describe a grammatical phenomenon.

3 Query Language

Our goal is to create syntactic patterns that reflect grammatical phenomena, as taught in primary school education, in a formal language that could be machine-readable, by using the dependencies of words in a sentence, and also adequately user-friendly. Our syntax should be able to map the dependencies among two or more words, use syntactic features (parts-of-speech, dependency labels), morphosyntactic features (lemma, case, number etc.) and orthographic features (one-on-one match with a word, punctuation etc.), and also be position-independent, so that it can find dependencies that span across the sentence.

Our approach is based on the theory of *abstract role values* of *constraint-based dependency grammars* (White, 2000). These grammars possess a set of *lexical categories* of the elements of a phrase, a set of their *roles*, a set of their *labels*, and these sets are governed by a set of *constraints* (Nivre, 2005). In our approach, we create sets of possible syntactic features for each word of the phrase separately (set of part-of-speech tags, set of dependency labels, set of morphosyntactic information) that should match the features of a word in a dependency parse. Then, we pair these sets with the sets of features that the word's head should possess (if a head-dependent connection is needed in the parser), and add more sets of features or tuples of dependent-head features if needed by the pattern. By *head*, we are referring to the head of a two-word phrase, not to the *root* of the sentence; this will allow us to build patterns referring to one-word rules or rules with words that are not directly dependent on the *root*.

We developed an extendable structure to cater to simple and complex structures. The first word that needs to be matched in a pattern is called *comp_word*, after the term *complement* in a head-driven phrase structure. This may have a set of possible parts-of-speech, labels, morphosyntactic features, lemmata, word forms, and morphemes. The second word is the *head_word*, the head of the first word as defined by the dependency parse. This one also has its own set of possible features, and the pattern will only be valid if both words are matched. A pattern template is presented in Figure 1.

comp_word: **POS**={A,B}&label={c}&
feature={d,e}&**lemma**={'e'}&
wordform={'f','g'}&
wordform={h-,i-}&
wordform={-j-},
head_word: **POS**={K}&**label**={l,m}&
feature={o}&**wordform**={-p,-q},
tokenID(head_word) = headID(comp_word)

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

Every field may have one or more possible values. The fields **POS**, **label**, **lemma**, and **wordform** will be matched with one of the corresponding features of the word. The field **feature** requires all listed morphosyntactic features of the pattern to match the morphology of the word. Not all possible sets need to be filled, as shown in the *head_word* features; the pattern can include as much relevant information as needed in each grammatical phenomenon. Values should be separated by a comma in every set, and brackets should be used when a word is used, e.g. in **lemma** and **wordform**. Concerning **wordform**, this field can contain either a specific word (preferably inflected), one or multiple prefixes, one or multiple suffixes, or one or multiple infixes. Different types of values should exist in their own **wordform** field, as demonstrated in *comp_word*.

In order to understand better how patterns are created and match words, we will examine a pattern to find a simple noun phrase with a definite article, in Figure 2.

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.

In order for this pattern to exist in a sentence or phrase, we need to have a word that is a *determiner* as part-of-speech, labeled as *determiner* by the dependency parser, have the features of *definiteness* and being an *article*, and be dependent to a word that is a *noun*. For example, this pattern would be found in the German sentence, *Die Katze schläft.* "The cat sleeps.". According to the dependency tree of the sentence in Figure 3 and the parse in Figure 1, there is a word matching the dependent (*Die*) and its head (*Katze*) matches the *head_word* of the pattern. Therefore, the pattern, and the rule for noun phrase, will be found.

This structure can also support simpler grammatical rules that only require matching one element of the sentence. All the fields that were used above can also be applied here. The word to be matched is tagged as *head_word*, as there is no dependency to create a *head-complement* set, e.g. a one-word pattern grammatical rule that looks for the presence of a definite article (regardless of its dependencies) shown in Figure 4. This pattern would be found in the previous example sentence, because the word *Die* matches all these requirements.

In order to describe more composite grammatical structures, we can use multiple syntactic patters of one or two words, combined. All separate patterns should be matched with the words in the sentence, in order of this compound syntactic pattern to be matched. Since in dependency parsing there is always a pair of *head-complement* no longer than two words, in order to describe phenomena that involve more

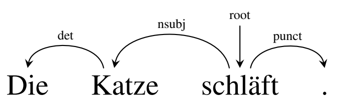

Figure 3: Dependency tree of the sentence *Die Katze schläft.*

Die	**Katze**	**schläft**	**.**
"der"	"Katze"	"schlafen"	.
POS=DET	POS=NOUN	POS=VERB	POS=PUNCT
label=det	label=nsubj	label=root	label=punct
Case=Nom	Case=Nom	Number=Sing	
Definite=Def	Gender=Fem	Person=3	
Gender=Fem	Number=Sing	VerbForm=Fin	
Number=Sing			
PronType=Art			
head="Katze"	head="schläft"		head="schläft"

Table 1: A CDG parse of the sentence *Die Katze schläft.*

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

Figure 4: Pattern to identify a definite determiner.

than two words, first we make patterns of one or two words and connect these patterns by finding their common denominator. This has to be a unique word in the utterance on which all the other words are *dependent*– the *root*. For example, in order to create a pattern for a simple sentence with a mono-transitive verb, e.g. *Er liebt Maria.* "He loves Maria.", our course of action would be to create a pattern that matches a *nominal subject* with a *verb* which is the *root* of the sentence, and a second pattern which matches a *direct object* with a *verb* that is also the *root* of the sentence. In a sentence, only one *root* should exist. Therefore, both patterns have the same *head_word*.

As shown in Figure 6 and Table 2, the compound pattern in Figure 5 will match the sentence 'Er liebt Maria', because both patterns in the compound pattern are matched.

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)

Figure 5: Pattern for a simple mono-transitive sentence.

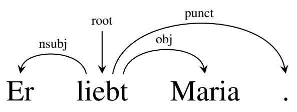

Figure 6: Dependency tree of sentence *Er liebt Maria.*

Er	**liebt**	**Maria**	**.**
"er"	"lieben"	"Maria"	.
POS=PRON	POS=VERB	POS=PROPN	POS=PUNCT
label=nsubj	label=root	label=obj	label=punct
Case=Nom	Number=Sing		
Gender=Masc	Person=3		
Number=Sing	VerbFrom=Fin		
Person=3			
head="liebt"		head="liebt"	head="liebt"

Table 2: CDG parse of the sentence *Er liebt Maria.*

Our previous pattern only used dependency labels and part-of-speech tags for a good reason; in the grammar rule we defined, we are looking for sentences with a nominal subject and a direct object, regardless of their part-of-speech (pronoun, a noun, a proper noun etc.) and their morphosyntactic features. However, this under-defining could prove problematic. Suppose we have a *reflexive sentence*, e.g. *Ich wasche mich.* "I wash myself." (Figure 8 and Table 3). This is a reflexive sentence, because the object of the sentence has the same reference as the subject. Reflexivity is a more complex syntactic structure than a simple sentence with two different entities as subject and object, and we would like to create a special pattern for reflexive sentences. See a pattern for such cases of simple reflexive sentences in Figure 7.

comp_word: **label**={nsubj},
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)

Figure 7: Pattern for a simple reflexive sentence.

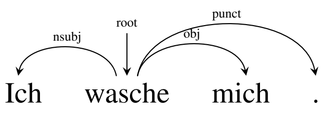

Figure 8: Dependency tree of sentence *Ich wasche mich.*

Ich	**wasche**	**mich**	**.**

“ich"
POS=PRON
label=nsubj
Case=Nom
Gender=Masc
Number=Sing
Person=1
PronType=Prs
 	
head=“wasche"
	“waschen"
	POS=VERB
	label=root
	Number=Sing
	Person=3
	VerbFrom=Fin
		“ich"
		POS=PRON
		label=obj
		Case=Acc
		Gender=Masc
		Number=Sing
		Person=1
		PronType=Prs
		Reflex=Yes
		head=“wasche"
			.
			POS=PUNCT
			label=punct
			
			head=“wasche"

Table 3: CDG parse of the sentence *Ich wasche mich.*

The sentence *Ich wasche mich.* would match the reflexive sentence pattern, but it would also match the aforementioned pattern for simple mono-transitive sentences, because in dependency parsing, reflexive pronouns are dependent on the head of the clause and not on the entity they reference. While this reflexive sentence is a mono-transitive sentence and the mono-transitive sentence pattern correctly matches it, we would like to keep these two structures separate from each other because of their different difficulties. We could add a constraint to the pattern for reflexive sentences that would state that if both the pattern for mono-transitive sentences and reflexive sentences is matched, then the most ‘relevant’ one is reflexive sentences. However, this approach would be difficult as our set of grammar rule patterns grows and we would have to keep track of all pre-existing possible matching patterns. Our second option would be to revise the way we define simple patterns and add *exclude operators* that would prevent more complex cases to be matched with simpler patterns. An exclude operator (tilde and parentheses) is wrapped around a pattern or a simple pattern and can be used in one or more patterns in a compound pattern. If the pattern inside the exclude operator is found, then the pattern is deemed to not be a match. For example, we would revise our simple mono-transitive sentences pattern to exclude the presence of an indirect object (hence, not matching bi-transitive sentences and the presence of reflexivity) in Figure 9.

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))

Figure 9: The revised pattern for simple mono-transitive sentences.

While this approach may seem more arduous, since we would have to take into account multiple cases when making a pattern, it enables the definition of very specific patterns that cater to specific grammatical phenomena, like this case of reflexive sentences. It can also help us define differences between patterns that cannot be taken account by using only dependencies. For example, a simple yes-no question in German, e.g. *Hast du Zeit?* “Do you have time?" (Figure 10) would have the same dependency structure as the sentence *Du hast Zeit.* “You have time." (Figure 11). Therefore, it is not possible to discern between these two cases with a pattern, unless we use an exclude operator that excludes the presence of a specific punctuation mark. The reason we are not using the positions of words in a sentence in our patterns for this case or any other pattern so far is because dependencies are meant to capture deep structural relationships, regardless of position. Declaring strict positions for arguments in a pattern could be problematic for languages that allow even small liberties in word order such as German. *Das Buch lese ich!* and *Ich lese das Buch!* both translate to “I read the book!" despite the surface structures being OVS and SVO, respectively. Ultimately, the choices on how patterns will match grammatical rules and sentences belong to the creators of the patterns for each language.

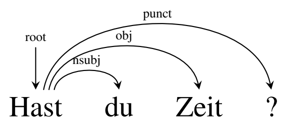

Figure 10: Dependency tree of *Hast du Zeit?*

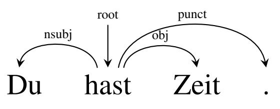

Figure 11: Dependency tree of *Du hast Zeit.*

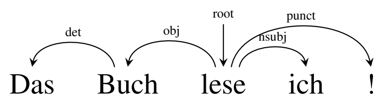

Figure 12: Dependency tree of *Das Buch lese ich!*

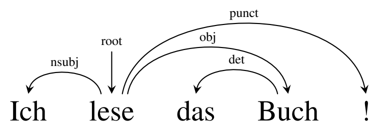

Figure 13: Dependency tree of *Ich lese das Buch!*

4 The matching process

4.1 Building syntactic patterns for German

Now that we have defined our query language, we will present the process of collecting the appropriate grammar rules and creating the patterns to find these rules in a sentence-level. Since our target demographic was primary school children, we had to focus on simpler grammar rules and syntactic structures, and pay close attention to what difficulty level we will assign to them, so that students would be introduced to concepts with a gradual difficulty and based on already acquired rules. It is important to understand which syntactic phenomena are used at each age. While Kauchak et al. (2007) have mentioned that the frequency of a parse tree structure correlates to its difficulty, there are more factors to how difficult a grammar rule is, e.g. young German students are not introduced to complex cases such as passive voice in German until 10/11 years old (*Klasse 5/6*). (Note that Germany does not have a unified school curriculum and syllabus, and every state defines their own standards; we consulted the school curricula of the German states of Saarland and Rheinland-Pfalz to understand which syntactic phenomena are used at each age.) To further study the syntactic phenomena, we consulted linguistics textbooks (Altmann and Hahnemann, 2007).

As was discussed in Section 3, we built the patterns following grammar rules as close as possible, excluding cases where the pattern would be too general. We used the Universal Dependencies 2.3 annotation schema for our patterns (Nivre et al., 2018a). So far, we have created 135 patterns for morphosyntactic and syntactic rules in German with their syntactic categories, a human-readable description, a difficulty score, and their prerequisite rules (a list of what rules need to be already known in order for this rule to be taught. It is used by our partners in the project to automatically curate content according to the user's level.). We present an abridged version of a few of syntactic rules, their difficulty, and the patterns we have created to match them; Table 4 with simple rules, Table 5 with complex rules and Table 6 with compound rules.

ID	Description	Dif.	Pattern
218	Auxiliary verb "sein", present indicative	1	head_word: POS={AUX}&wordform={"bin","bist","ist","sind","seid","sein"}&feature={VerbForm=Fin}
222	Auxiliary verb "haben", present indicative	1	head_word: POS={AUX}&wordform={"hab","habe","hast","hat","haben"}&feature={Mood=Ind,VerbForm=Fin}

Table 4: A few simple syntactic patterns to match one word. 'Dif' is the assigned difficulty.

ID	Description	Dif.	Pattern
240	Composed forms: Perfect indicative	1	comp_word: {<222>,<218>}, head_word: POS={VERB}&feature={VerbForm=Part}, tokenID(head_word)=headID(comp_word)
261	Adjective is Predicate to Noun	1	comp_word: POS={NOUN,PROPN,PRON}, head_word: POS={ADJ}&label={root}, tokenID(head_word)=headID(comp_word)
281	Two-part Coordinate conjunctions	2	comp_word: POS={CCONJ}&label={cc}, head_word: POS={CCONJ}&label={cc}, tokenID(head_word)=headID(comp_word)
287	Prepositions with accusative	2	comp_word: POS={ADP}&label={case}, head_word: POS={NOUN,PROPN}&feature={Case=Acc}, tokenID(head_word)=headID(comp_word)

Table 5: A few complex syntactic patterns for one dependent word (261) or a dependent word and its head (240, 281, 287). Note that the complement side of rule 240 is the simple rules 222 or 218 from Table 4.

ID	Description	Dif.	Pattern
288	Simple clause with intransitive verb	1	(comp_word: label={nsubj}, head_word: POS={VERB}&label={root}, tokenID(head_word)=headID(comp_word)) AND ∼(head_word: label={obj}) AND ∼(head_word: label={iobj}) AND ∼(head_word:POS={VERB}&label={root}&feature={VerbForm=Part}) AND ∼(head_word: POS={PUNCT}&wordform={"?"}) AND ∼(head_word: feature={Mood=Imp}&label={root})
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})
290	Simple clause with transitive verb	1	(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 ∼(head_word: label={iobj}) AND ∼(head_word:POS={VERB}&label={root}&feature={VerbForm=Part}) AND ∼(head_word: POS={PUNCT}&wordform={"?"}) AND ∼(head_word: label={obj,iobj}&feature={Reflex=Yes}) AND ∼(head_word: feature={Mood=Imp}&label={root})
292	Simple clause with bitransitive verb	2	(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 ∼(head_word: POS={VERB}&label={root}&feature={VerbForm=Part}) AND ∼(head_word: POS={obj,iobj}&feature={Reflex=Yes}) AND ∼(head_word: POS={PUNCT}&wordform={"?"}) AND ∼(head_word: feature={Mood=Imp}&label={root})
294	Reflexive sentence with transitive verb	1	(comp_word: label={nsubj}, head_word: POS={VERB}&label={root}, tokenID(head_word)=headID(comp_word)) AND (comp_word: label={obj,iobj}&feature={Reflex=Yes}, head_word: POS={VERB}&label={root}, tokenID(head_word)=headID(comp_word)) AND ∼(head_word:POS={VERB}&label={root}&feature={VerbForm=Part}) AND ∼(head_word: POS={PUNCT}&wordform={"?"}) AND ∼(head_word: feature={Mood=Imp}&label={root})
296	Simple clause with predicate	1	(comp_word: label={nsubj}, head_word: POS={ADJ}&label={root}, tokenID(head_word)=headID(comp_word)) AND (comp_word: POS={VERB,AUX}&label={cop}, head_word: POS={ADJ}&label={root}, tokenID(head_word)=headID(comp_word)) AND ∼(head_word: POS={PUNCT}&wordform={"?"}) AND ∼(head_word: feature={Mood=Imp}&label={root})
298	Simple clause with separable verb	2	(comp_word: POS={ADP}&label={compound:prt}, head_word: POS={VERB}, tokenID(head_word)=headID(comp_word)) AND ∼(head_word: POS={PUNCT}&wordform={"?"}) AND ∼(head_word: feature={Mood=Imp}&label={root})
299	Simple w- question (yes-no)	1	(comp_word: label={nsubj}, head_word: POS={VERB}&label={root}, tokenID(head_word)=headID(comp_word)) AND (comp_word: POS={PUNCT}&wordform={"?"}, head_word: label={root}, tokenID(head_word)=headID(comp_word)) AND ∼(head_word: feature={PronType=Int}) AND ∼(head_word: POS={VERB}&label={root}&feature={VerbForm=Part}) AND ∼(head_word: feature={Mood=Imp}&label={root})
301	Simple w- question where adverb/pronoun is Subject	1	(comp_word: POS={PRON}&label={nsubj}&feature={Case=Nom,PronType=Int}, head_word: POS={VERB}&label={root}, tokenID(head_word)=headID(comp_word)) AND (comp_word: POS={PUNCT}&wordform={"?"}, head_word: label={root}, tokenID(head_word)=headID(comp_word)) AND ∼(head_word: feature={Mood=Imp}&label={root})
302	Simple question, adverb or pronoun is Complementizer	3	(comp_word: label={advmod}&feature={PronType=Int}, head_word: label={root}, tokenID(head_word)=headID(comp_word)) AND (comp_word: POS={PUNCT}&wordform={"?"}, head_word: label={root}, tokenID(head_word)=headID(comp_word)) AND ∼(head_word: feature={Mood=Imp}&label={root})

Table 6: A few of the compound patterns that will match an entire (simple) clause.

4.2 Dictionaries and the case of multi-word expressions

The patterns need to be matched with a parsed sentence. We use the *MUNDERLINE* parser of (Volokh and Neumann, 2012) for dependency parsing in CoNLL-U trees with Universal Dependencies annotation tags. However, we require some further linguistic information for some patterns, which cannot be provided from a parser, e.g. morphemes. As part of the German language resources, we have created a dictionary of 15,000 German words, from our 117K corpus of age-appropriate texts: children's texts from children's magazines and newspapers (*GEOlino*[1], *GEOlino Extra*[2], *Dein SPIEGEL*[3]), children's

[1]`https://www.geo.de/geolino`

[2]`https://www.geo.de/magazine/geolino-extra`

[3]`https://www.deinspiegel.de/`

literature (*Phontasia*[4]) and pedagogical material (works of *Ursula Rickli*[5]). These words are annotated with lemma and stem information, their phonological and morphological features, their morphemes and orthographic syllables. Our dictionary is relatively small, because children's texts tend to be simple, but it should cover the most important words for sentences relevant to German primary school students.

Concerning multi-word expressions (MWEs), there are still open discussions on how they should be handled (Gerdes and Kahane, 2016). Even so, parsers are usually unable to identify them. For example, while 'as well as' is considered a fixed MWE, parsers tend to fail to capture the dependencies between the words of the MWE and the MWE's head. Our approach to identifying multi-word expressions and making use of them in syntactic patterns was to reduce a multi-word expression post-parse to a single phrase retaining the features of only one word of the MWE (Kato et al., 2016). It was only marginally worse than training a parser with MWE awareness (Candito and Constant, 2014). For example, in the sentence "We like John as well as Mary.", the MWE *as well as* is a coordinate conjunction between *John* and *Mary* (Figure 14 and Table 7). The first *as* has the features of the coordinate conjunction, and *well* and *as* are the subsequent words dependent to *as*. We concatenate the MWE as one word, *as well as*, into one word, retaining only the features of the first *as*, since it is the head of the MWE.

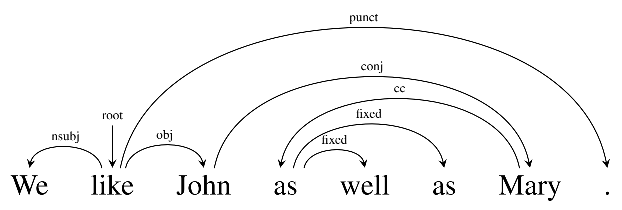

Figure 14: Dependency tree of sentence "We like John as well as Mary."

John	as	well	as	Mary
"John" POS=PROPN label=obj head='like'	"as" POS=ADV label=cc head='Mary'	"well" POS=ADV label=fixed Degree=Pos head='as'	"as" POS=ADP label=fixed head='as'	"Mary" POS=PROPN label=conj head='John'

Table 7: Part of the CDG parse for the sentence "We like John as well as Mary.". Note that the parsers we tested were not able to successfully annotate 'well' and 'as' with 'fixed'– this is a gold parse.

In order to recognize MWEs and which one of their components is syntactically important (whose features we are going to annotate the joined MWE with), we compiled a list of the MWEs we deem relevant for primary school level texts. This allows us to expand if necessary and handle other languages of the project.

4.3 Matching algorithm

By using our syntactic patterns for German and the required resources (dictionary, list of MWEs), we now proceed to build an algorithm to match patterns with dependency parses. In Section 3, we presented three different types of patterns: one-word patterns (Fig. 4), two-word patterns (Fig. 2) and compound patterns (Fig. 5, 7, 9). In order to match these rules to a parse, we first transform the parse from CoNLL-U format to a dictionary of dictionaries, where every word of the sentence has its own dictionary of features (part-of-speech, dependency label, head, morphosyntactic features). We also read the patterns we have made to a dictionary. Our goal is to check how many patterns matched our sentence's words, which patterns matched, and what their dependencies are (if applicable). We have created three algorithms which step-by-step look up every word in the sentence and try to match it (and its head, if applicable) to patterns. First, it tries to match the word's features with features of simple rules (one-word patterns). Then, it tries to match the word with the complement side of a complex rule. If the complement side is a match, it finds the head of the word (if applicable) and tries to match the head word's features with the head word's features in the complex rule. Finally, for rules composed of multiple patterns (compound rules), it tries to match every pattern of the rule, simple or complex, by nesting the algorithms described above.

[4]http://www.phontasia.de/
[5]http://www.ursularickli.ch/

5 Results and Discussion

From our corpus mentioned in Section 4.2, we created two sets: a development set of 152 sentences, which we used during the process of creating the syntactic patterns to fine-tune them, and a test set of 101 sentences to test the performance of our patterns and matching algorithm. We created their dependency parses manually, as a gold standard, and annotated them with the ideal grammar rules that they should be matched with. First of all, we would like to present a few sentences from the test set with the annotated matches and the matches that the algorithm returned with the use of gold standard parses, in Table 8. Some of the patterns for the rules can be found in Tables 4, 5 and 6. The matcher also returns the position of the *head_word* and the *comp_word* if applicable. As shown, the matches are mostly correct for all three types of patterns, meaning that our query language and our patterns are robust enough to describe and find the syntactic phenomena that we aimed to identify.

Sentence	Gold standard rule		Matched rule
Die Reise hat mehrere Tage gedauert. "The journey took several days."	205	Function words – Definite article	205
	269	Discourse anaphors – NP with definite article	269
	222	Function words – Auxiliary verb "haben", present indicative	222
	217	Function words – Indefinite pronouns	217
	240	Morphosyntax – Composed forms: Perfect indicative	240
	289	Clause structure – Simple clause, intransitive verb, with auxiliary verb	289
Ihre Mutter hat eine gute Idee. "Your mother has a good idea."	208	Function words – Possessive pronoun, Nominative	208
	273	Discourse anaphors – NP with possessive pronoun	273
	206	Function words – Indefinite article	206
	270	Discourse anaphors – NP with indefinite article	270
	260	Adjectives – Attribute to noun	260
	290	Clause structure – Simple clause, transitive verb	290
Jetzt konnte sich die Raupe am Ästchen festhalten. "Now the caterpillar could hold on to the branch."	307	Adverbs	307
	230	Function words – Auxiliary verbs, past	230
	274	Binding – Reflexive pronouns	274
	205	Function words – Definite article	205
	269	Discourse anaphors – NP with definite article	269
	294	Clause structure – Reflexive sentence, transitive verb	294
	233	Function words – Prepositions with dative	
	286	Discourse anaphors – Prepositional phrase in dative	
Du bist aber schick. "But you are chic."	212	Function words – Personal pronouns, nominative	212
	265	Discourse anaphors – Personal pronouns as Subject	265
	238	Function words – Particles	238
	307	Adverbs	307
	261	Adjectives – Predicate	261
	296	Clause structure – Simple clause with predicate	296
Ein Schiff fährt auf dem Meer entlang. "A ship sails along the sea."	206	Function words – Indefinite article	206
	270	Discourse anaphors – NP with indefinite article	270
	235	Function words – Prepositions with accusative and dative	235
	205	Function words – Definite article	205
	269	Discourse anaphors – NP with definite article	269
	284	Discourse anaphors – Separable prefix	284
	286	Discourse anaphors – Prepositional phrase in dative	286
	288	Clause structure – Simple clause, intransitive verb	288
	298	Clause structure – Simple clause, with separable verb	298
Hattest du denn keine Arbeit? "Did you have no work then?"	212	Function words – Personal pronouns, nominative	212
	265	Discourse anaphors – Personal pronouns as Subject	265
	238	Function words – Particles	238
	307	Adverbs	307
	207	Function words – Negative Indefinite article	207
	271	Discourse anaphors – NP with negation	271
	299	Clause structure – Simple question (yes-no)	299
	283	Negation	283
Wer steht da neben deinem Vater? "Who stands there next to your father?"	216	Function words – Interrogative pronouns	216
	275	Discourse anaphors – Interrogative pronoun, determiner, numeral or adverb	275
	307	Adverbs	307
	235	Function words – Prepositions with accusative and dative	235
	286	Discourse anaphors – Prepositional phrase in dative	286
	301	W-clauses – Simple question, adverb or pronoun is Subject	301

Table 8: Seven sentences from our test set, their ideal matches and the matches that the algorithm returned. Rules that were incorrectly matched/not matched are marked in bold.

In addition, we used several dependency parsers to parse the sentences and then evaluated their CoNLL-U trees to our gold standard. We wanted to ensure that our parser would have adequate performance and wouldn't cause mismatches that could be avoided with the use of another parser. The parsers we chose are all either pre-trained or trained on the German GSD Universal Dependencies treebank (McDonald et al., 2013): *UDPipe* (Straka and Straková, 2017)), *jPTDP* (Nguyen and Verspoor, 2018) and *Turku neural parser pipeline* (Kanerva et al., 2018). The results for the development set can be found in Table 9, and for the test set in 10. *Turku* was the most successful of the parsers, creating more accurate CoNLL-U trees with significantly higher recall than *MUNDERLINE* and *UDPipe*. *jPTDP* performed poorly because it does not output morphosyntactic features (FEATS) in its CoNLL-U trees, which is an important input to the matcher. However, in our parse tree evaluations, *jPTDP* recreated UPOS, HEAD, and DEPREL columns at least as well as –if not better than– *Turku*. *jPTDP* can be executed on top of CoNLL-U trees with FEATS, but for the purposes of our experiment, we only considered direct output of our input of a list of sentences for each parser.

	Gold Standard	*MUNDERLINE*	*UDPipe*	*jPTDP*	*Turku*
Total	978	978	911	978	911
TP	922	742	638	249	788
FP	26	92	105	98	89
FN	56	236	273	729	123
Precision	0.9726	0.8897	0.8587	0.7176	**0.8985**
Recall	0.9427	0.7587	0.7003	0.2546	**0.8650**
F1	0.9574	0.8190	0.7715	0.3758	**0.8814**

Table 9: Matcher results on the development set with gold standard parses and parse results from the parsers.

	Gold Standard	*MUNDERLINE*	*UDPipe*	*jPTDP*	*Turku*
Total	776	776	664	776	664
TP	734	601	496	246	587
FP	32	80	89	68	96
FN	42	175	168	530	77
Precision	0.9582	**0.8825**	0.8479	0.7834	0.8594
Recall	0.9459	0.7745	0.7470	0.3170	**0.8840**
F1	0.9520	0.8250	0.7942	0.4514	**0.8716**

Table 10: Matcher results on the test set with gold standard parses and parse results from the parsers.

At first glance, it may seem odd that the matcher was not able to match 100% of the rules corresponding to the gold standard. This may be due to our choices on the way we built the patterns; we did not aim for a complete representation of the German language and all the possible expressions of a grammar rule because our goal was to successfully match sentences with grammatical phenomena that are taught in primary school. Therefore, if a sentence has a grammatical rule that is expressed in a way not covered by our patterns, the pattern will not be matched. For example, the sentence *Jetzt konnte sich die Raupe am Ästchen festhalten.* (Table 8) contains the prepositional phrase *am Ästchen*. Even though there is a syntactic pattern to match prepositional phrases with the dative case, the pattern is not matched because it requires a noun as the head of the phrase, and it does not support substitution, which is a more complex syntactic phenomenon. However, the sentence is annotated with the grammar rule because the rule is present.

Additionally, the way that dependency parsing expresses some structures may cause some matches to not occur. For example, in the sentence, *Alles war grün und gelb.* "Everything was green and yellow.", only *grün* will be matched with the rule for predicate (261, 296), because *grün* is labeled as *adjective* which is the *root* of the sentence, but *gelb* is correctly labeled as *conjunct* to *grün*, because they are connected with a conjunction. Even though they have the same syntactic role, conjunct parts of speech cannot be matched to patterns that require a specific label. In the future, we will consider ways to overcome this problem, for example by adding rules to add enhanced dependencies as described in (Nivre et al., 2018b).

We also noticed that problems occur in prepositional phrases when the preposition and the determiner are contracted to one word. Ideally, in German treebanks they are analysed to preposition and determiner, but parsers overall failed to decompose these contractions. For example, in the sentence *Rotkäppchen musste zum Hause gehen.* "Little Red Riding Hood had to go home.')", *zum Hause* would be decomposed to *zu dem Hause*, and then the rule for prepositional phrases with dative would be found. Since parsers are not analysing the contraction, the pattern will not be matched. Multi-word tokens like *zum* caused differences in parse trees for UDPipe and Turku compared to our gold standard. In those instances, we excluded them from the results, which is why the total number of features matched for the two parsers is lower than the other parsers in Tables 9 and 10.

6 Future work

Our future work will be to extend the matcher to other languages. Our partners are working on creating syntactic patterns for grammar rules on a primary education level for English, Greek and Spanish. We would like to assess the pattern quality and the matcher performance for other languages as we did for German. Since our matcher is language-independent and Universal Dependencies includes annotations that cover the majority of documented languages, we are optimistic that we will have satisfactory results. Additionally, we would like to integrate the matcher to the text difficulty metric that has been developed by our other partners, since our patterns correspond to grammar rules with annotated difficulty.

As part of our ongoing research, we would like to further explore how the matcher and the syntactic patterns could be used in other NLP applications. We would like to solve problems such as conjunctions and contractions, and eventually graduate to more complex patterns. For example, we would like to create patterns for analysis of more complex sentences, something that could easily be achieved since our matcher successfully recognizes simple sentences in a clause (e.g. the matcher will return the rule for a subordinate clause and for a simple sentence, in the case of a conditional sentence).

Acknowledgements

We would like to thank our reviewers for their valuable insight. This work was partially funded by the European Union's Horizon 2020 grant agreement No. 731724 (iREAD).

References

Hans Altmann and Suzan Hahnemann. 2007. *Syntax fürs Examen: Studien-und Arbeitsbuch*, volume 1. Vandenhoeck & Ruprecht.

Waleed Ammar. 2016. *Towards a Universal Analyzer of Natural Languages*. Ph.D. thesis, Google Research.

Franck Bodmer. 1996. Aspekte der Abfragekomponente von COSMAS II. *LDV-INFO*, 8:142–155.

Marie Candito and Matthieu Constant. 2014. Strategies for contiguous multiword expression analysis and dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 743–753.

Kim Gerdes and Sylvain Kahane. 2016. Dependency annotation choices: Assessing theoretical and practical issues of universal dependencies. In *Proceedings of the 10th Linguistic Annotation Workshop held in conjunction with ACL 2016 (LAW-X 2016)*, pages 131–140.

Jenna Kanerva, Filip Ginter, Niko Miekka, Akseli Leino, and Tapio Salakoski. 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.

Akihiko Kato, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Construction of an English Dependency Corpus incorporating Compound Function Words. In *LREC*.

David Kauchak, William Coster, and Gondy Leroy. 2012. A Systematic Grammatical Analysis of Easy and Difficult Medical Text. In *AMIA*.

David Kauchak, Gondy Leroy, and Alan Hogue. 2017. Measuring text difficulty using parse-tree frequency. *Journal of the Association for Information Science and Technology*, 68(9):2088–2100.

Tobias Kuhn and Stefan Höfler. 2012. Coral: Corpus access in controlled language. *Corpora*, 7(2):187–206.

Wolfgang Lezius. 2002. TIGERSearch—ein Suchwerkzeug für Baumbanken. In *Proceedings der 6. Konferenz zur Verarbeitung natürlicher Sprache*, volume 6, pages 107–114.

Scott Martens. 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*.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 92–97.

Dat Quoc Nguyen and Karin Verspoor. 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, October. Association for Computational Linguistics.

Joakim Nivre, Mitchell Abrams, and Željko Agić et al. 2018a. Universal dependencies 2.3. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Joakim Nivre, Paola Marongiu, Filip Ginter, Jenna Kanerva, Simonetta Montemagni, Sebastian Schuster, and Maria Simi. 2018b. Enhancing Universal Dependency Treebanks: A Case Study. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 102–107.

Joakim Nivre. 2005. Dependency grammar and dependency parsing. *MSI report*, 5133(1959):1–32.

Petr Pajas and Jan Štěpánek. 2009. System for querying syntactically annotated corpora. In *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, pages 33–36. Association for Computational Linguistics.

Adam Przepiórkowski, Zygmunt Krynicki, Lukasz Debowski, Marcin Wolinski, Daniel Janus, and Piotr Banski. 2004. A Search Tool for Corpora with Positional Tagsets and Ambiguities. In *LREC*.

Milan Straka and Jana Straková. 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, August. Association for Computational Linguistics.

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

Christopher M White. 2000. Rapid grammar development and parsing: Constraint dependency grammars with abstract role values. *West Lafayette, Indiana, USA: PhD Thesis, Purdue University*.

Amir Zeldes, Anke Lüdeling, Julia Ritz, and Christian Chiarcos. 2009. ANNIS: A search tool for multi-layer annotated corpora.