

Teaching grammar visually

Interactive visualization of syntactic structure assembly for grammar-oriented first- and second-language instruction

Gerard Kempen and Camiel van Breugel

Cognitive Psychology Unit, Leiden University

Abstract

Understanding the grammatical structure of the target language is beneficial to certain categories of language learners, in particular to students learning to write in a language with syntax-sensitive spelling rules, and to L1/L2 learners who prefer explicit rules. Moreover, grammar is an important body of knowledge in its own right.

Two visual-interactive grammar instruction tools are presented that graphically support students in composing and transforming sentences. The structure of sentences is displayed in the form of easily interpretable syntactic trees that the student can assemble and modify interactively. The tools check on-line the grammatical well-formedness of the structures being manipulated. The programs are based on the new Performance Grammar formalism (e.g., Kempen and Harbusch 2002). They have been implemented in JAVA and can run under most modern operating systems.

1 Introduction

Few foreign-language teachers and learners consider grammar their favorite subject matter. Whether or not explicit grammatical knowledge helps attaining *oral* proficiency in a second language, is controversial. Many learners become fluent speakers and smooth understanders simply by practicing these skills, without having paid much attention to the grammar rules of the target language. However, this observation does not justify downplaying the importance of grammar in the acquisition of *written* language skills, neither in the mother tongue nor in a foreign language. Striking examples are provided by languages with large numbers of homophonous wordforms whose orthography is dictated by syntactic context. Writers in these languages, e.g. French and Dutch, are likely to commit numerous errors if they ignore syntax-sensitive spelling rules (Pijls, Daelemans and Kempen 1987; Kempen and Dijkstra 1994; Kempen 1999).¹

Grammar rules underlying the syntax-sensitive aspects of verbform spelling tend to be rather abstract, and difficult for primary and secondary schoolers to understand and apply. Consequently, negligible or even negative effects of explicit grammar instruction on L1 and L2 acquisition are unavoidable. This pessimistic conclusion, which is probably valid in the context of grammar instruction (if any) as currently practiced in many schools, has become a self-fulfilling prophecy. It has abated the interest of linguists and educationists in designing improved grammar didactics. The revolutionary developments in the cognitive sciences (linguistics, informatics, psychology) over the past decades indeed have hardly affected

¹This paper is an extended version of Kempen (2004).

the teaching of grammar (Kempen 1996). We venture the prediction that explicit grammar instruction can boost the proficiency gains in L1 and L2 curricula by profiting from recent developments in the cognitive sciences, in particular computational linguistics.

In this paper, we describe two interactive software tools for the on-line visualization and manipulation of syntactic structures. The first program is intended as an exercise tool in L1 courses on grammatical terminology (“sentence analysis”). The students can train the application of grammatical concepts by constructing easily understandable phrase-structure trees for sentences prepared by the teacher. The tool checks the correctness of the student responses on-line. The second program lets students compose new L2 sentences on the computer screen by assembling syntactic trees from partial phrase-structure trees (“treelets”) associated with individual words. The tool immediately displays on-line the morphological and syntactic consequences of any assembly operation performed by the student. It refuses assembly attempts that would yield ungrammatical results and can provide feedback on the reason for the refusal. It thus guides the student to the construction of grammatically correct L2 sentences.

Section 2 outlines some basic properties of the syntactic processor, the functionality of the tools, the user interfaces, and the status of the current implementations. Positive results of initial user evaluations are mentioned in Section 3.

2 Syntactic processor and user interfaces

2.1 Performance Grammar

Both tools are based on the Performance Grammar formalism (PG) developed by Kempen and Harbusch (Kempen and Harbusch 2002; Harbusch and Kempen 2002; Kempen and Harbusch 2003). Because PG trees have n -ary branching nodes, they tend to be “flat”—flatter than, e.g., the binary right-branching trees typical of Generative Grammar. Moreover, PG makes systematic use of functional concepts such as Subject, Direct Object, Head, etc., that are also used in traditional school grammars. Syntactic movement operations (e.g., “raising transformations”) do not leave “traces” and are displayed graphically in a way that transparently brings out the source and target positions of moved constituents. Every word of the language is the head of a so-called lexical frame, that is, a treelet whose root can be merged (technically: unified) with a foot node of treelets associated with other words. This enables constructive exercises where students build sentences not by stringing words together but by assembling trees that automatically apply morpho-syntactic (hierarchical, word order) constraints. These features qualify PG as a formalism suitable for use in grammar teaching for linguistic novices.

2.2 PGW and PGT

The current computer implementation of PG is called the Performance Grammar Workbench (PGW). It covers a range of grammatical constructions of Dutch sufficiently wide to serve as the linguistic engine of a grammar teaching tool. Its

graphical layer includes advanced tree drawing and manipulation functions capable of displaying “elastic” phrase-structure and dependency trees whose branches can be selected via mouse clicks and reordered from left to right without distorting the horizontal and vertical alignment of the nodes of the resulting tree.

Students can launch treelets by dragging words from a lexicon window to a workspace. Here, treelets and trees-under-construction can be picked up (on “mouse-down”) and moved around freely. As soon as a root node hits upon a non-lexical foot node of another tree(let), the system checks whether or not the nodes are “unifiable”, given the current configuration of morpho-syntactic features in the two structures. If so, the unification is realized when the student decides to drop (“mouse-up”) the treelet at the current position. Otherwise, the node merger is refused and the tree(let) is dropped at a nearby position. If unification does take place, the system computes any morpho-syntactic implications for the shape of the extended tree and for the features of its node, and the new tree is drawn afresh. Any unification can be undone by picking up a branch of a treelet and pulling it away from the tree it belongs to. The shape of the trimmed tree and of the detached treelet as well as the features on their nodes are recomputed and set to the state that would have resulted if the undone unification had never taken place before.

The second program, called Performance Grammar Trainer (PGT), may be viewed as a “light” version of the PGW. It does not incorporate a linguistic engine and can only display static PG trees. Teachers can define new sentence analysis (“parsing”) exercises by graphically constructing syntactic trees with three types of nodes: word classes (parts of speech), word groups (phrases), and grammatical functions. Once stored, these trees serve as the criterion of correctness for students who are performing parsing exercises. Teachers can tune the PGT to the terminological preferences of their grammar curriculum. Moreover, they can adapt the trees to the proficiency level of the student by selecting which type(s) of nodes is/are displayed on the screen. Due to this flexibility, the program can be used as an add-on to many printed grammar curricula and is independent of the native language of the students.

2.3 Current implementations

Both the PGW and the PGT have been written in JAVA and run under most modern operating systems, either as autonomous programs or as applets within internet browsers. The sequence of screenshots in Figures 1–6 (see Appendix) show some of the functionality and the user interfaces of the PGT. A PGT exercise starts either with a “flat” presentation of the to-be-analyzed sentence or with a tree whose node labels are missing. The latter alternative provides “scaffolding” and helps the student to become familiar with tree diagrams. At each labeling attempt, the student is free to select one of four types of units: an individual word, a continuous string of words, an individual grammatical label, or a continuous string of labels. After that, s/he selects one option from one of three pop-up menus, one for each type of nodes. The PGT automatically positions the selected label in the tree. Errors can be restored at any time. The order in which the labels are assigned, is arbitrary.

The remaining figures of the Appendix are PGW screenshots. Figures 7–12 depict a PGW exercise. Here, the program helps an L2 student of Dutch to compose sentence (1) and its synonymous variant (2) with Subject–Verb inversion.

- (1) Je ziet hier een magnifiek theater
 you see here a magnificent theater
 ‘Here you see a magnificent theater’
- (2) Hier zie je een magnifiek theater
- (3) Je had een magnifiek theater kunnen zien
 you would-have a magnificent theater been-able-to see
 ‘You would have been able to see a magnificent theater’

The graphical component of the PGW can display discontinuous constituency in the form of directed graphs. In example (3), the Direct Object NP of the embedded clause headed by *zien* ‘see’ has moved upward into the main clause (as a consequence of “clause union”). This movement is indicated by the dashed lines in Figure 13. Furthermore, notice the possibility of toggling between various types of sentence diagrams. Phrase-structure trees, which are the default, can automatically be converted to dependency trees. Both types of syntactic trees are in use in modern grammar teaching projects that heavily rely on visual representations of syntactic structure. For instance, the Danish VISL project uses *n*-ary phrase-structure trees (URL: www.visl.sdu.dk) while R. Hudson prefers a notation based on dependency trees (URL: www.phon.ucl.ac.uk/home/dick/tta/KS3.htm; see also Barton and Hudson 2003). Finally, as illustrated by Figures 14–16, the level of detail of the syntactic annotations at the nodes of dependency trees can be tailored to the knowledge and preferences of the students.

3 Evaluation and conclusion

In each of two formal evaluation experiments, 18 undergraduate Dutch-language university students used the PGT during two 45-minute periods in order to refresh their high-school knowledge of grammatical concepts. The students were enthusiastic about the tool and produced considerable learning gains in this relatively short period of time. At the time of writing, no formal user evaluation of the PGW is available.

Despite the small amount of practical experience with the tools, we conclude that PGW and PGT represent a proof of concept showing that useful tools for grammar-oriented L1 and L2 instruction can emerge from the combined technologies of natural language processing, dynamic visualization, and direct-manipulation user interfaces. We expect that thanks to innovative tools of this sort grammar-oriented language instruction methods will ultimately secure their legitimate position next to the currently dominant communicative didactics.

Acknowledgement

We are indebted to Robert Berg for implementing the PGT and for his help in running one of the evaluation experiments. We also thank Freek Gertsen and Nard Loonen for constructing the PGT website for students of the Hogeschool Arnhem-Nijmegen (HAN) in Nijmegen.

References

- Barton, G. and Hudson, R. (2003). Diagrams, *Times Educational Supplement*, April 4th, 2003.
- Harbusch, K. and Kempen, G. (2002). A quantitative model of word order and movement in English, Dutch and German complement constructions, *Proceedings of the 19th International Conference on Computational Linguistics (COLING-2002)*, Taipei (Taiwan), Morgan Kaufmann, San Francisco (California), pp. 328–334.
- Kempen, G. (1996). Human Language Technology can modernize writing and grammar instruction, *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, Center for Sprogteknologi, Copenhagen, Copenhagen (Denmark), pp. 1005–1006.
- Kempen, G. (1999). Visual Grammar: Multimedia for grammar and spelling instruction in primary education, in Cameron, K. (ed), *CALL: Media, design, and applications*, Swets and Zeitlinger, Lisse (The Netherlands), pp. 223–238.
- Kempen, G. (2004). Interactive visualization of syntactic structure assembly for grammar-intensive first- and second-language instruction, in Delmonte, R., Delcloque, Ph. and Tonelli, S. (eds), *Proceedings of InSTIL/ICALL2004 Symposium on NLP and speech technologies in advanced language learning systems*, University of Venice, Venice (Italy), pp. 183–186.
- Kempen, G. and Dijkstra, A. (1994). Toward an integrated system for grammar, writing and spelling instruction, in Appelo, L. and de Jong, F.M.G. (eds), *Computer-Assisted Language Learning. Proceedings of the Seventh Twente Workshop on Language Technology*, University of Twente, Enschede (The Netherlands), pp. 41–46.
- Kempen, G. and Harbusch, K. (2002). Performance Grammar: A declarative definition, in Nijholt, A., Theune, M. and Hondorp, H. (eds), *Computational Linguistics in the Netherlands 2001*, Rodopi, Amsterdam, pp. 148–162.
- Kempen, G. and Harbusch, K. (2003). Dutch and German verb constructions in Performance Grammar, in Seuren, P. and Kempen, G. (eds), *Verb constructions in German and Dutch*, Benjamins, Amsterdam, pp. 185–221.
- Pijls, F., Daelemans, W. and Kempen, G. (1987). Artificial Intelligence tools for grammar and spelling instruction. *Instructional Science*, 16, pp. 319–336.

Appendix: Screenshots from PGT (Figures 1–6) and PGW (Figures 7–16)

sentence

The rain in Spain stays mainly in the plain

Figure 1: Flat initial presentation of a sample sentence.

Function	Phrase	Word class
subject	subclause	noun
direct object	noun phrase	adjective
indirect object	adjectival phrase	verb
prepositional object	prepositional phrase	article
head of predicate	adverbial phrase	preposition
nominal part of predicate	verb phrase	numeral
verbal rest of predicate		pronoun
modifier		adverb
subordinator		conjunction
particle		

Figure 2: Menu options (three types of nodes).

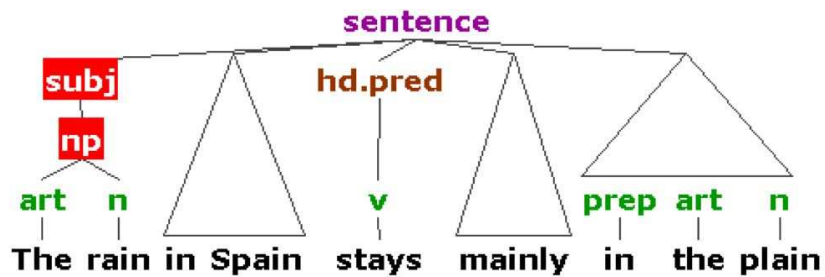


Figure 3: An intermediate solution stage with errors in Subject NP.

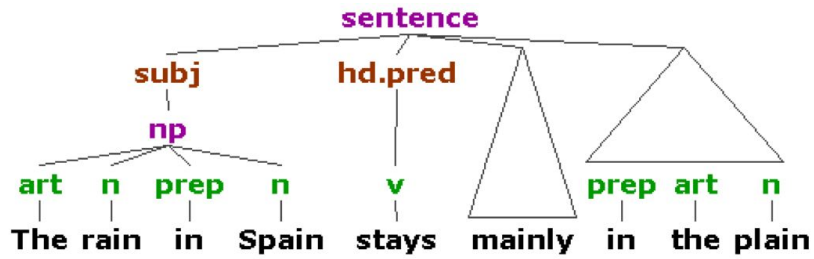


Figure 4: Intermediate stage with errors corrected.

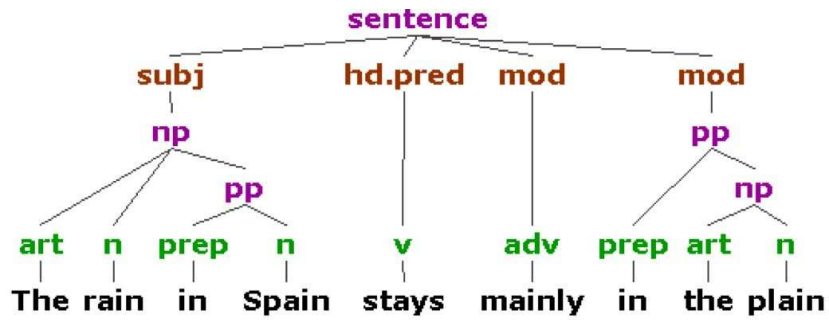


Figure 5: Final correct analysis. Notice that, in line with current practice in most Dutch schools, grammatical function labels are restricted to the level of main and subordinate clauses. Moreover, phrase labels are inserted only for phrases longer than one word. Both conventions aim to reduce the complexity of the trees for beginning learners.

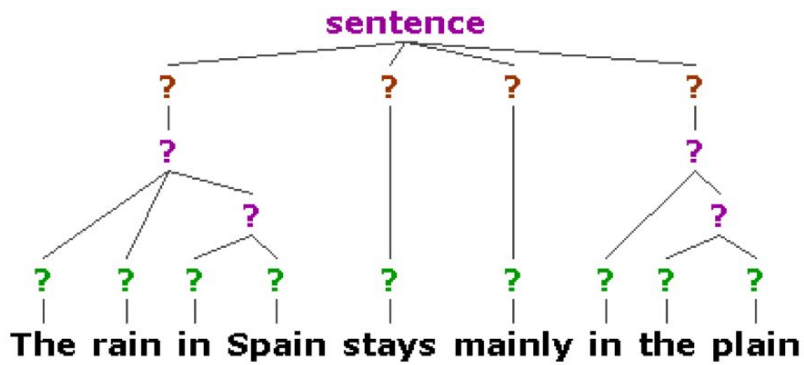


Figure 6: Alternative initial presentation of the PGT exercise.

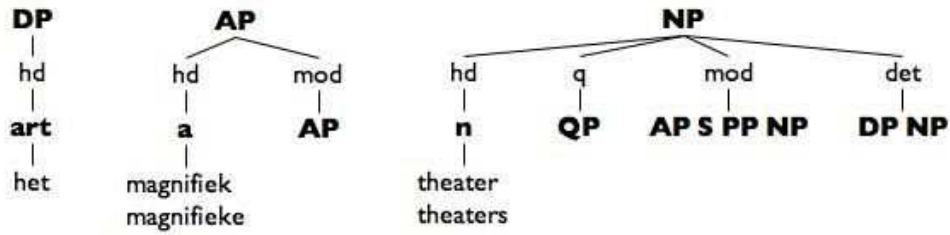


Figure 7: Three treelets retrieved from the PGW lexicon. Alternative wordforms are printed below the head branch.

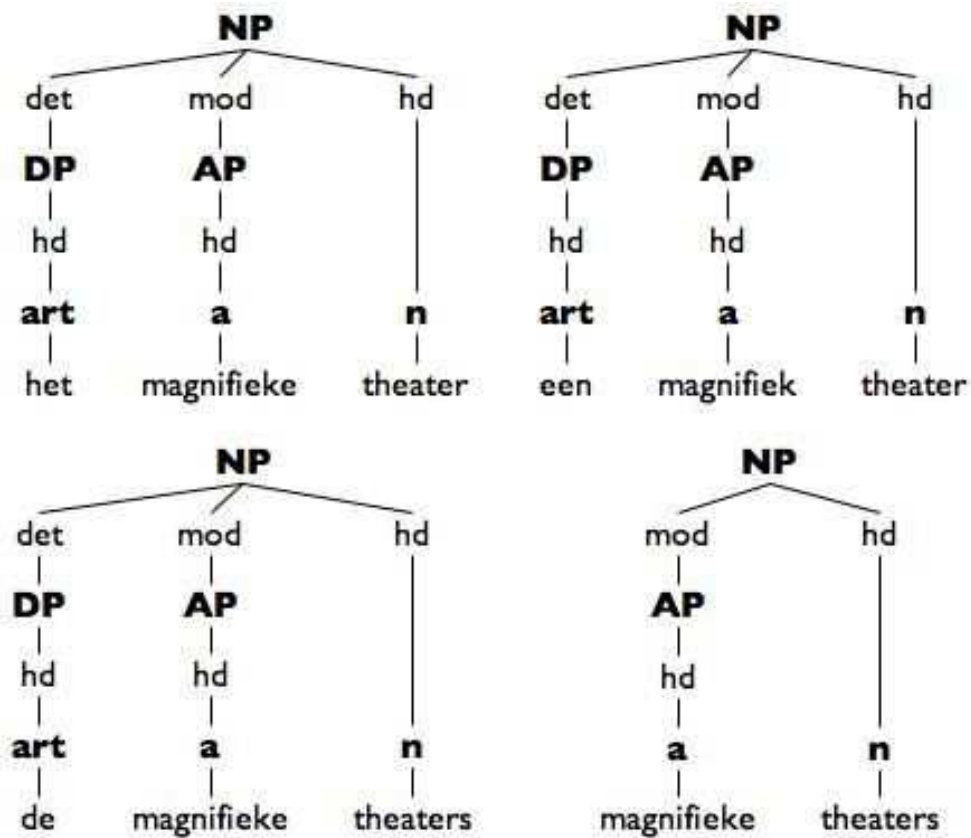


Figure 8: Upon unification, PGW's linguistic engine automatically selects the correct word-form(s) and word order(s). (*De, het* and *een* are articles.) Notice inflection of adjective and noun.

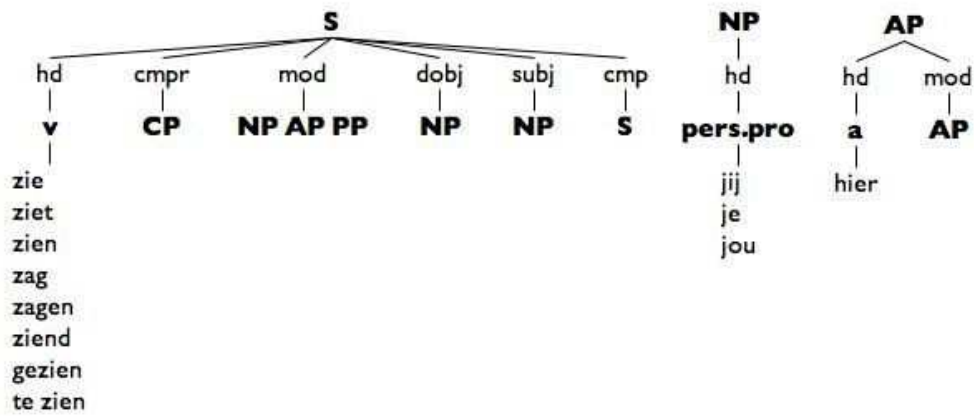


Figure 9: Treelets for the remaining words of sentences (1) and (2).

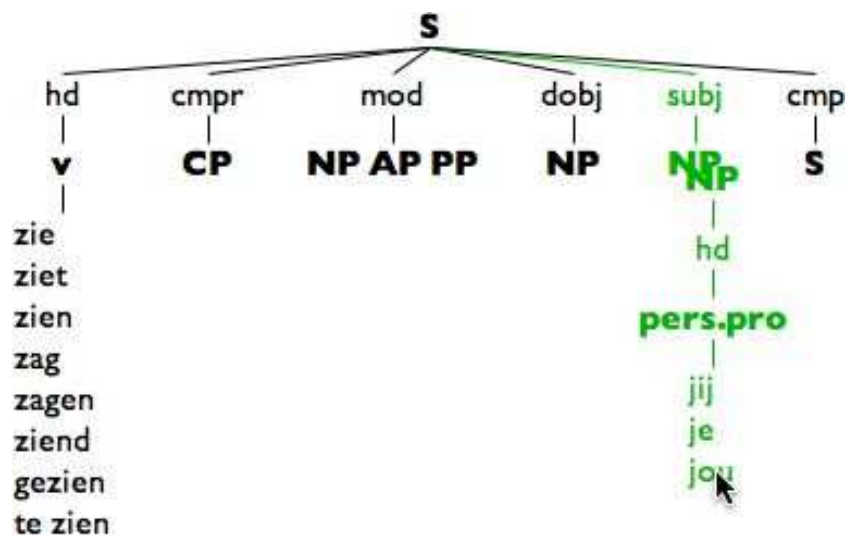


Figure 10: In order to test whether two treelets can be combined, the student drags the root of one tree(let) over the target foot node of another.

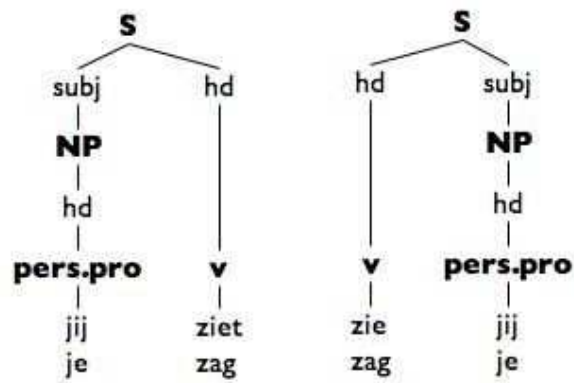


Figure 11: The configuration in the previous figure triggers a unification attempt by the PGW. Here unification succeeds, resulting in a reduced set of applicable wordforms and two ordering options. (Some branches have been pruned away for reasons of clarity.)

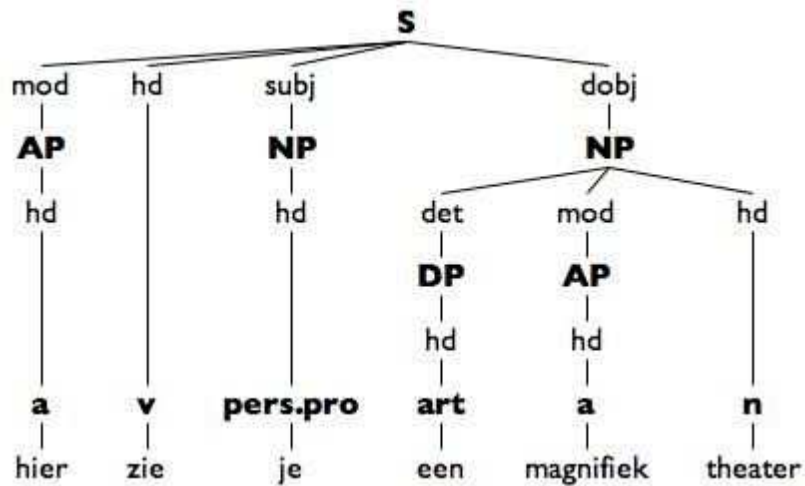


Figure 12: Dragging the root NP of *een magnifiek theater* over the Direct Object foot node of the verb yields sentences (1) and (2). This is the tree for (2), after superfluous branches have been pruned away, and without the alternative wordforms *jij* 'you' and *zag* 'saw'. Sentence (1) is identical, except that Subject and Modifier have been interchanged and that the verbforms differ.

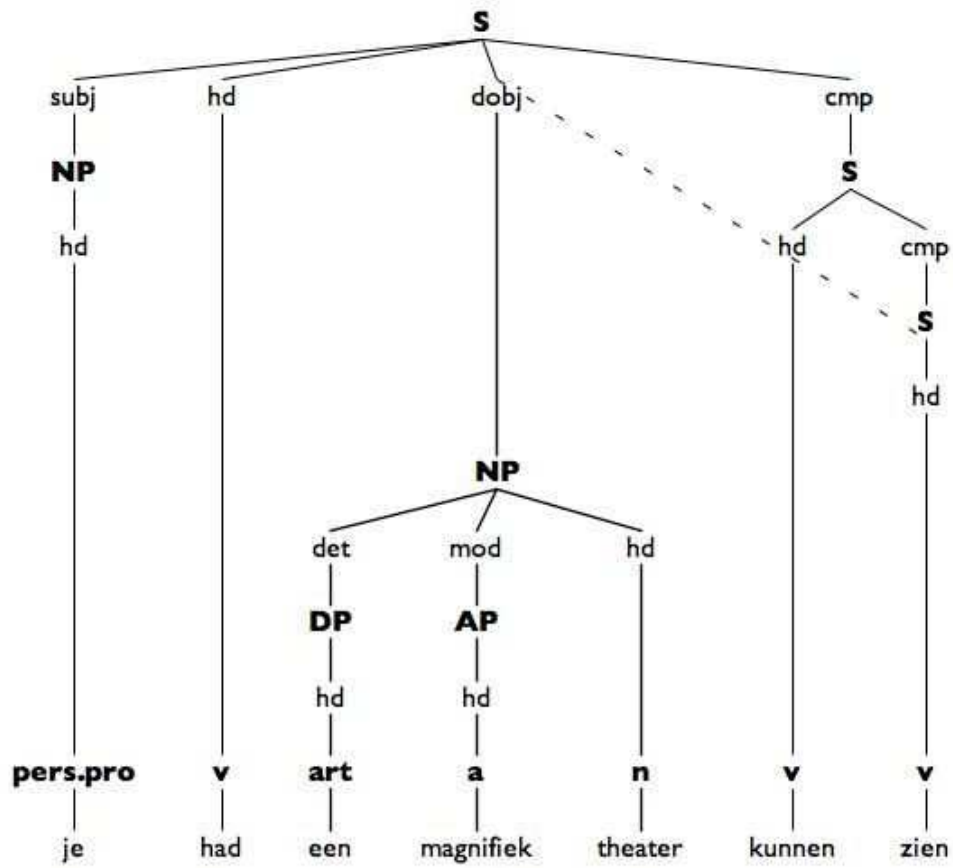


Figure 13: A tree-like diagram for sentence (3) with a discontinuous constituent. The dashed line shows upward movement of the Direct Object NP from the most deeply embedded clause into the main clause.

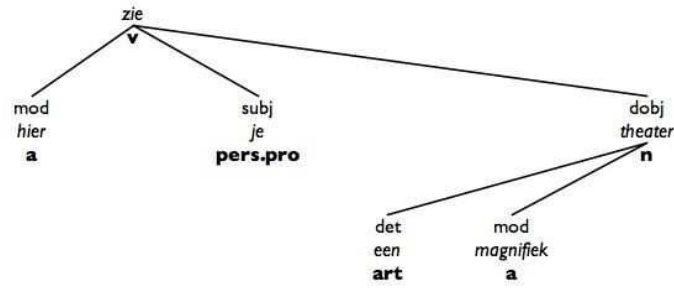


Figure 14: Dependency tree for sentence (2). Lexical nodes are annotated by grammatical function and part-of-speech labels.

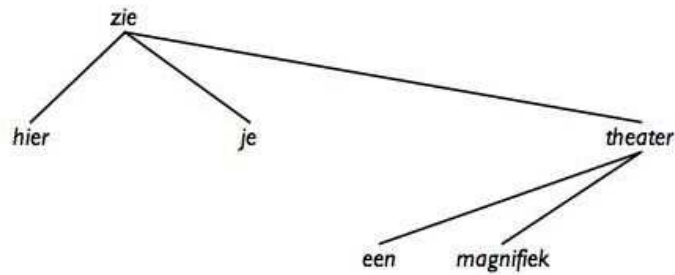


Figure 15: Dependency tree for sentence (2) without any syntactic annotation.

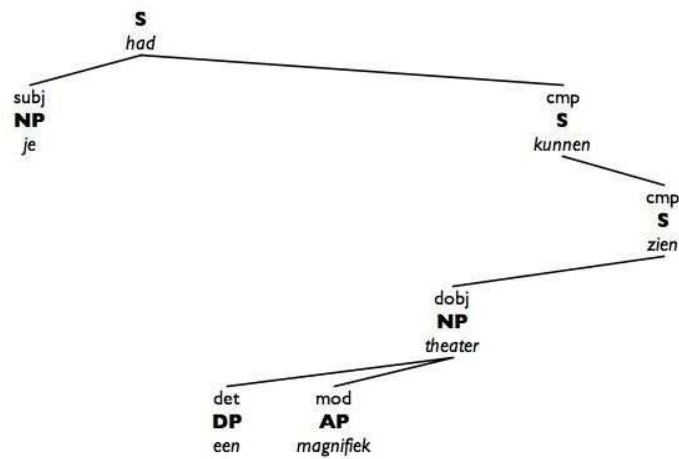


Figure 16: Dependency tree for sentence (3) annotated with grammatical function and phrase labels.