Gapping Constructions in Universal Dependencies v2

Sebastian Schuster
October 24, 2017

1 Introduction

An important property of human language is that speakers can sometimes omit redundant material. One example of this phenomenon is so-called gapping constructions (Ross 1970). In such constructions, speakers elide a previously mentioned verb that takes multiple arguments, which leaves behind a clause without its main predicate. For example, in the sentence "*John likes tea, and Mary coffee*", the verb *likes* was elided from the second conjunct. In this paper, I consider all constructions in which a perdicate that has multiple dependents was elided, including classic cases of gapping (Ross 1970). Throughout this paper, I call the elided material (a predicate and occasionally also some of its arguments) the GAP. Further, I refer to the dependents of the gap as ORPHANS or REMNANTS, and I refer to the dependents of the predicate in the clause with the overt predicate as the CORRESPONDENTS, as illustrated with the following annotated sentence.

John	likes	tea	and	Mary	coffee
CORRE-SPONDENT	OVERT VERB	CORRE-SPONDENT		ORPHAN/ REMNANT	ORPHAN/ REMNANT
FULL CLAUSE				GAPPED CLAUSE	

Sentences with gapping pose practical as well as theoretical challenges to natural language processing tasks. From a practical point of view, it is challenging for natural language processing systems to identify and resolve the gaps, which is necessary to interpret these sentences and extract information from them. For example, if one wants to automatically extract all the events from the following sentence from a news article, the system has to identify that there are four elided *killing*-events, and that each of the conjuncts contains the subject and a locational modifier of each of these events.

(1) Seven were killed in Damietta, four in Ismania, three in Giza, three in Cairo, and one in Sohag.

Further, these sentences are also hard for statistical parsers to parse as part of their structure deviates significantly from canonical clause structures.

From a more theoretical point of view, these constructions pose challenges to designers of dependency representations. Most dependency representations that are used in natural language processing systems (e.g., Surdeanu et al. 2008, de Marneffe et al. 2006, Nivre et al. 2016) are concerned with providing surface syntax descriptions without stipulating any additional transformations or empty nodes. Further, virtually all dependency representations consider a verb (either the inflected or the main verb) to be the head of a clause. Consequently, the verb governs all its arguments and modifiers. For these reasons, it is challenging to find a good representation of clauses in which a verb that has multiple dependents was elided, because it is not obvious where and how the remaining dependents should be attached in these cases.

In recent years, the Universal Dependencies (UD) representation (Nivre et al. 2016) has become the dominant dependency representation for annotating treebanks in a large variety of languages. The goal of the UD project is to provide guidelines for cross-linguistically consistent treebank annotations for as many languages as possible. Considering that gapping constructions appear in many languages, these guidelines necessarily also have to include guidelines on how to analyze gapping constructions. While the official guidelines[1] provide basic instructions for the analysis of gapping constructions, they lack a detailed discussion of cross-linguistically attested gapping constructions and a thorough explanation why the adopted guidelines should be preferred over other proposals. The purpose of the present paper is therefore to compare the adopted analysis of gapping to another analysis that was considered in the development of the second version of the guidelines.

1.1 Coordination and ellipsis in UD v2

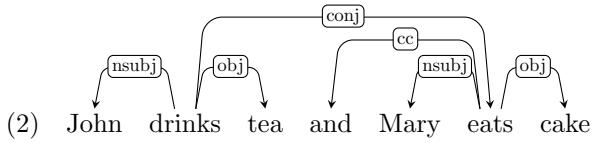
Before I discuss the proposals for analyzing gapping constructions in UD v2, I give a brief overview of how UD analyzes coordinated clauses and other forms of elliptical constructions.

Coordinated clauses are analyzed like all other types of coordination: By convention, the head of the first conjunct is always the head of the coordinated construction and all other conjuncts are attached to the head of the first conjunct with a `conj` relation. If there is an overt coordinating conjunction, we[2] attach it to the head of the succeeding conjunct. This captures the fact

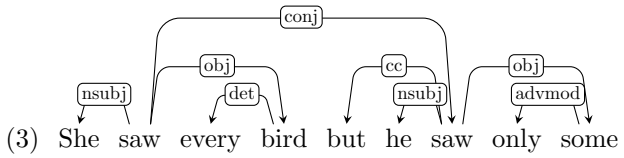
1 See http://universaldependencies.org/u/overview/specific-syntax.html#ellipsis.

2 As I am part of the team who designs the UD guidelines, I use "we" to refer to "the people who work on/within the UD framework".

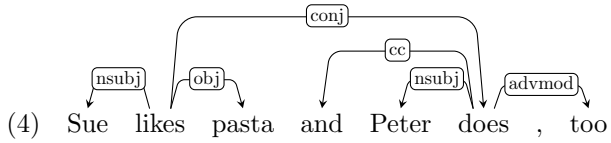
that the coordinating conjunction forms a syntactic unit with the succeeding conjunct (Ross 1967, Gerdes & Kahane 2015). A sentence with two coordinated clauses is then analyzed as follows.



For constructions in which a head nominal was elided, UD promotes the highest dependent according to the hierarchy `amod` > `nummod` > `det` > `nmod` > `case`. The promoted dependent is attached to the governor of the elided nominal with the same relation that would have been used if the nominal had not been elided. All the other dependents of the elided noun are attached to the promoted dependent with their regular relations. For example, in the second conjunct of the following sentence, the head noun *birds* was elided. One therefore promotes the determiner *some* to serve as the object of *saw*.



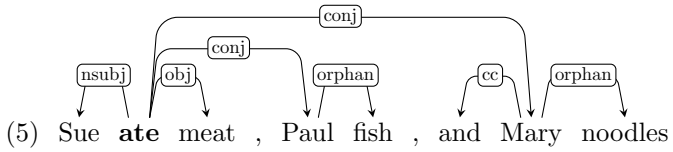
In some cases of ellipsis, a verb phrase is elided but there is still an overt copula or auxiliary verb. In these cases, the copula or auxiliary verb is promoted to be the head of the clause and all orphans are attached to the auxiliary.



1.2 Two proposals for analyzing gapping in UD

The constructions that I am primarily concerned with in this paper are gapping constructions in which the governor of multiple phrases was elided. As part of the discussion of version 2 of the UD guidelines, two analyses, the ORPHAN analysis and the COMPOSITE analysis, were considered.

Orphan analysis UD v2 ultimately adopted the ORPHAN analysis, which is a modified version of a proposal by Gerdes & Kahane (2015). According to this analysis, the orphan whose grammatical role dominates all other orphans according to an adaptation of the obliqueness hierarchy,[3] is promoted to be the head of the conjunct. The motivation behind using such a hierarchy instead of a simpler strategy such as promoting the leftmost phrase is that it leads to a more parallel analysis across languages that differ in word order. We attach all other orphans except for coordinating conjunctions using the special `orphan` relation. Coordinating conjunctions are attached to the head of the following conjunct with the `cc` relation. This leads to the analysis in (5) of a sentence with three conjuncts of which two contain a gap.



The motivation behind using a special `orphan` relation is that it indicates that the clause contains a gap. If one used a regular relation, it might not be clear that a predicate was elided. For example, if instead of using the `orphan` relation, we attached the orphaned clausal subject *making pizza* in (6) to the object using the canonical `csubj` relation, one could confuse gapping constructions with copular constructions,[4] especially in languages with zero-copula.

(6) Baking bread requires a very hot oven and making pizza an even hotter one.

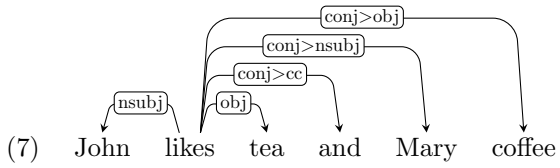
Composite analysis Joakim Nivre and Daniel Zeman developed a second proposal[5] as part of the discussion of the second version of the UD guidelines. Their proposal is based on composite relations such as `conj>nsubj`, which

3 UD's adaptation prioritizes phrasal over clausal dependents. Translated to UD relations, the adaptation of the obliqueness hierarchy is as follows: `nsubj` > `obj` > `iobj` > `obl` > `advmod` > `csubj` > `xcomp` > `ccomp` > `advcl`. See, for example, Pollard & Sag (1994) for a motivation behind this ordering.

4 In order to achieve better cross-linguistic consistency, UD treats the complement of the copula as the head of a copular clause and all the clausal arguments and modifiers are attached to the complement of the copular verb. E.g., in a simple equative sentence such as "*She is my mother*", we treat *mother* as the head of the clause and attach the subject *she* to *mother* using a `nsubj` relation.

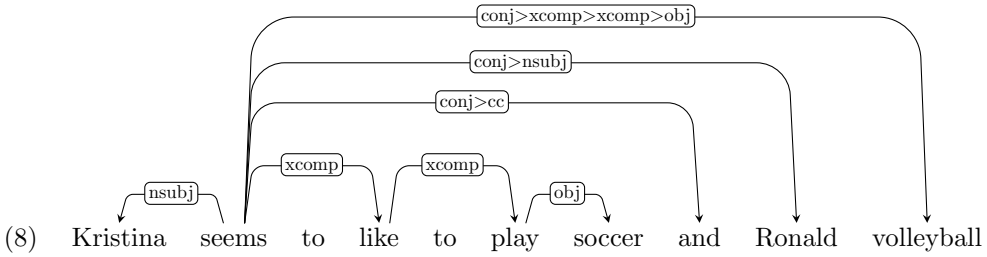
5 See http://universaldependencies.org/v2_prelim/ellipsis.html for a more detailed description of their proposal.

indicate which relations would be present along the dependency path from the first conjunct to the orphan if there was no gap. For example, `X conj>nsubj Y` indicates that there would have been a `conj` relation between X and an elided node, and an `nsubj` relation between the elided node and Y. According to this proposal, we would analyze a sentence with a single verb gap as follows.



The main motivation behind this proposal is that the relation names provide much more information on the type of dependent than the generic `orphan` relation, and that in most cases, it is straightforward to turn the gapped clause into a structure that looks more like a canonical clause structure.

Note that composite relations can also encode dependency paths of lengths greater than 2. For example, in (8), the relation label between the head of the full clause, *seems*, and the object remnant *volleyball* encodes a dependency path of length 4.



1.3 Design criteria in UD

In order to determine which of these two proposals is the better one, I first explain what constitutes a better analysis. The Universal Dependencies framework is not a syntactic theory and does not make predictions about grammaticality or acceptability, so it is impossible (and would not make sense) to assess the two proposals in terms of predictions of grammaticality judgements. Instead, Universal Dependencies should be seen as a dependency formalism in the sense of Mel'čuk (1988):

> Dependency formalism is a tool proposed for representing linguistic reality, and, like any tool, it may or may not prove sufficiently

> useful, flexible, or appropriate for the task for which it has been devised; but it cannot be true or false.
>
> (Mel'čuk 1988, p. 12)

Along the lines of what Mel'čuk calls "useful, flexible or appropriate for the task", UD defines the following six, often competing, goals and aims to find the best possible compromise between them.

> 1. UD needs to be satisfactory on linguistic analysis grounds for individual languages.
> 2. UD needs to be good for linguistic typology, i.e., providing a suitable basis for bringing out cross-linguistic parallelism across languages and language families.
> 3. UD must be suitable for rapid, consistent annotation by a human annotator.
> 4. UD must be suitable for computer parsing with high accuracy.
> 5. UD must be easily comprehended and used by a non-linguist, whether a language learner or an engineer with prosaic needs for language processing. We refer to this as seeking a *habitable* design, and it leads us to favor traditional grammar notions and terminology.
> 6. UD must support well downstream language understanding tasks (relation extraction, reading comprehension, machine translation, ...).
>
> (Christopher Manning, UD documentation[6])

Applied to my research question of determining the best analysis for gapping constructions, these goals translate approximately to the following questions.

1. a. Which analysis enables consistent analyses of gapping constructions within many languages?
   b. Which analysis assigns a linguistically reasonable structure to gapping constructions?

6 http://universaldependencies.org/introduction.html#what-is-needed-for-ud-to-be-successful

2. Which analysis allows for cross-linguistically consistent annotations of gapping constructions?
3. Which analysis makes it easier for humans to annotate gapping constructions?
4. Which analysis leads to annotations that can be automatically parsed with high accuracy?
5. Which analysis is easier to comprehend for a non-linguist?
6. Which analysis leads to a better representation for downstream language understanding tasks?

The ORPHAN analysis makes it arguably easier for humans to annotate gapping constructions as, unlike the case with the COMPOSITE analysis, a human annotator does not have to construct the sometimes very complex composite relations during the annotation process, so the ORPHAN analysis wins in terms of design goal 3. As for the comprehensibility of the two analyses (design goal 5), neither of them are likely to be understandable by a non-linguist without an explanation of what gapping constructions are.

The answers to the other questions seem less clear and answering them is the main content of this paper. In Section 2, I first look at gapping constructions from a theoretical point of view to answer questions 1 and 2. In Section 4, I present several parsing experiments to answer question 4. Finally, in Section 5, I try to answer the last question about the usefulness in downstream tasks.

2 Theoretical considerations

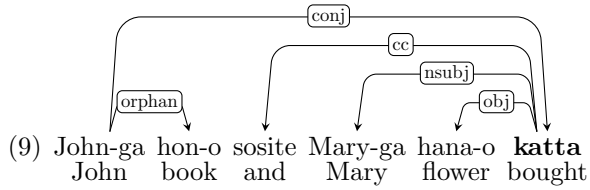
I first consider the questions whether both proposals can be used to consistently analyze attested gapping construction within and across languages. For that purpose, I discuss what kind of constructions with gapping are attested in the syntactic literature as well as in the English UD treebank (Silveira et al. 2014, Nivre et al. 2017a,b), and I show how these constructions can be analyzed according to the ORPHAN proposal.

2.1 Attested gapping constructions

2.1.1 Single verbs

The most common form of gapping constructions are two or more conjoined clauses in which a single inflected verb is missing in all but one of the conjuncts. As illustrated in (5), in SVO languages such as English, the overt verb typically

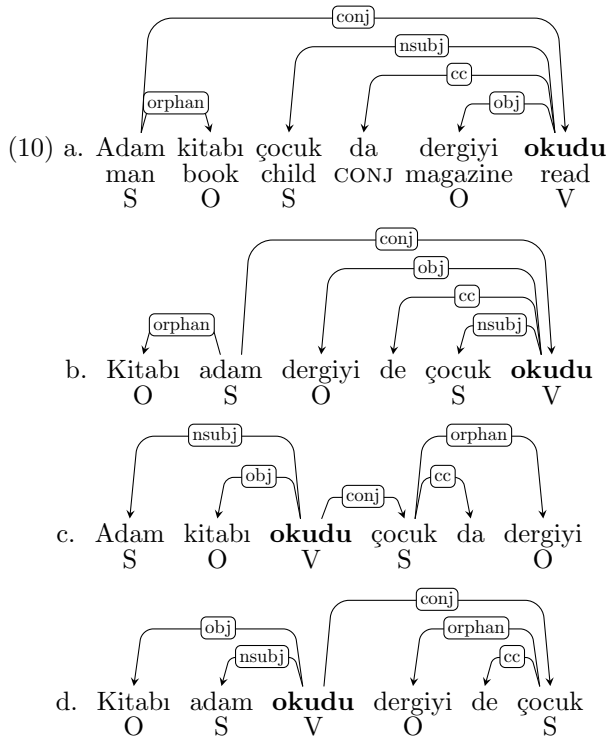
appears in the first conjunct and is elided from all subsequent conjuncts. In languages with other word orders, the overt verb can also appear exclusively in the last conjunct. For example, in the following sentence with gapping in Japanese (an SOV language), the verb appears in the last conjunct and the gap in the first conjunct.



'John bought books, and Mary flowers.'

(Kato 2006)

In some languages with flexible word orders such as Turkish, the overt verb can appear in the first or the last conjunct. The orphans typically appear in the same order as the correspondents in the other conjunct as in (10a-d) (Bozsahin 2000).

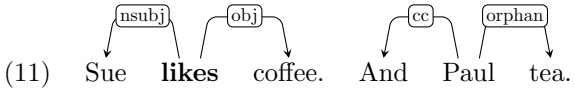


'The man read the book, and the child the magazine.'

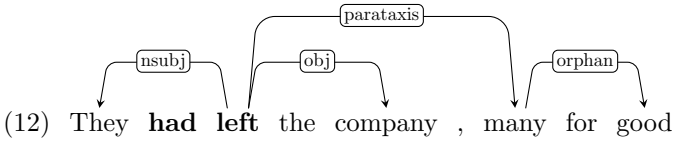
(Bozsahin 2000)

As mentioned above, in order to achieve higher cross-linguistic consistency, the first conjunct is always the head of a coordinated structure in UD. Therefore, the head of the first conjunct is always the head of the coordination, but the internal structure of each conjunct is the same for all four variants in (10).

In some cases, e.g., in certain discourse settings, the clause with the gap is not part of the same sentence as the clause with the overt verb (Gerdes & Kahane 2015). In these cases, we promote one of the orphans to be the root of the second sentence; the internal structure of the two clauses is the same as when they are part of an intra-sentential coordination.

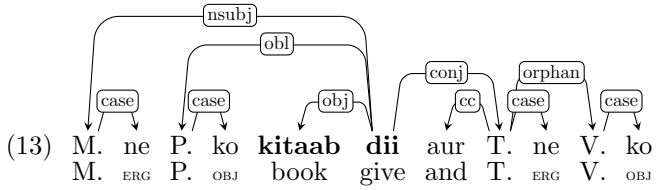


Further, conjuncts with a gap can also contain additional types of arguments or modifiers that are not present in the full clause. For example, in the sentence in (12), the oblique modifier *for good* does not correspond to any phrase in the first conjunct.



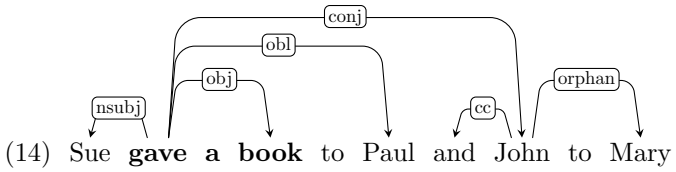
2.1.2 Verbs and their arguments or modifiers

Many languages also allow gapping of verbs along with some of their arguments or modifiers as illustrated in the following two examples in Hindi (13) and English (14).



'Manu gave a book to Pari and Tanu to Vimla'

(Kush 2016)



We analyze these cases as analogous to sentences in which only a verb was elided. The subject is promoted to be the head of the second conjunct and the oblique argument is attached with an `orphan` dependency.

2.1.3 Verbs and clausal complements

Ross (1970) points out that gaps can also correspond to a finite verb and one or more embedded verbs. For example, in the following sentence, it is possible to elide the matrix verb and all or some of the embedded verbs.

(15) I want to try to begin to write a novel, ...

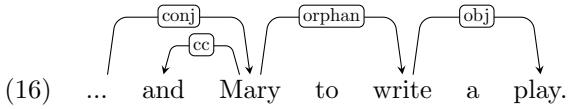
a. ... and Mary to try to begin to write a play.

b. ... and Mary to begin to write a play.

c. ... and Mary to write a play.

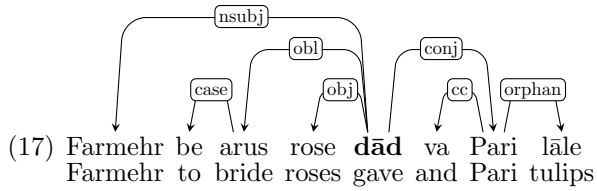
d. ... and Mary a play. (Ross 1970)

In all of these variants, the matrix verb was elided from the second conjunct. While this is an example of subject control and therefore *Mary* is also the subject of all the embedded verbs, it would be misleading to attach *Mary* to one of the embedded verbs because this would hide the fact that the matrix verb was elided. For this reason, we treat *Mary* as the head of the second conjunct and attach the remainder of the embedded clause with an `orphan` relation.



2.1.4 Non-contiguous gaps

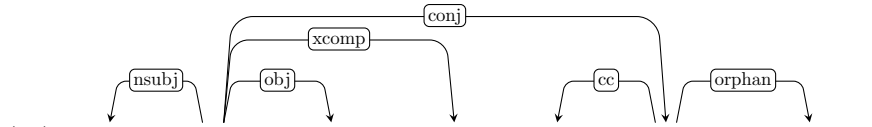
In the previous examples, the gap corresponds to a contiguous sequence in the first conjunct. However, as highlighted by the following examples, this is not always the case.



(17) Farmehr be arus rose **dād** va Pari lāle
Farmehr to bride roses gave and Pari tulips

'Farmehr gave roses to the bride and Pari (gave) tulips (to the bride)'

(Farudi 2013)



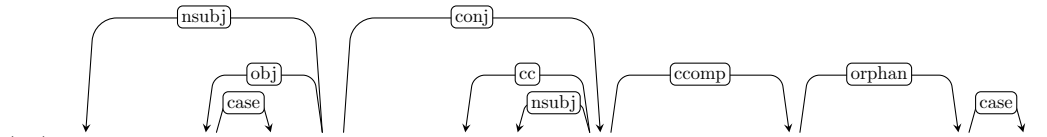
(18) Arizona **elected** Goldwater **Senator** , and Pennsylvania Schweiker

(Jackendoff 1971)

While the interpretation of the second conjunct is only possible if one fills both gaps, we are also in these cases primarily concerned with the elided verb because neither of the phrases in the second conjunct depend on the second gap. We can therefore analyze constructions with non-contiguous gaps in a similar manner as constructions with contiguous gaps, namely by promoting one orphan to be the head of the conjunct and attaching all other orphans to this head.

2.1.5 Gaps in embedded clauses

Farudi (2013) notes that in Farsi, gaps can appear in embedded clauses even if the corresponding verb in the first conjunct is not part of an embedded clause. For example, in (19), *dust* ('like') is the main verb of the highest clause of the first conjunct but in the second conjunct, the verb was elided from a clause embedded under *mi-dun-e* ('think').



(19) Ma. in ketāb ro **dust** dār-e va Mi. mi-dun-e ke māmān-esh un ketāb ro
Ma. this book OBJ like have and Mi. know that mother that book OBJ

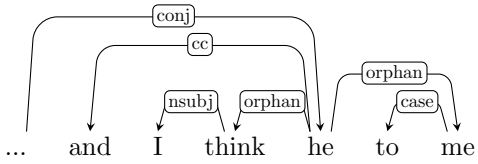
'Mahsa likes this book and Minu knows that her mother (likes) that book'

(Farudi 2013)

In these cases, we consider the matrix verb to be the head of the second conjunct as we would if there was no gap, and we promote the subject of the embedded clause (*māmān-esh*) to be the head of the embedded clause. We attach the remaining orphans to the subject with the `orphan` relation.

Note that this construction is different from constructions in which parenthetical material (Pollard & Sag 1994) appears in the second conjunct, as in the following English example.

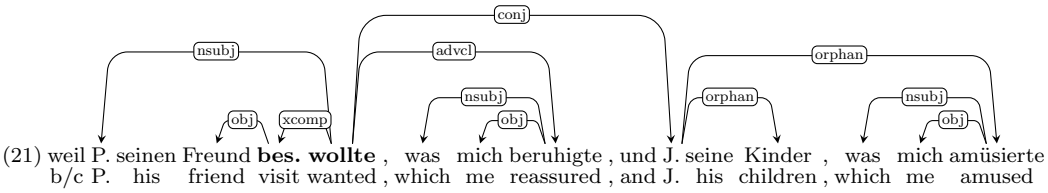
(20) [...] I always had a pretty deep emotional connection to him, and I think he to me.[7]



In these cases, we promote the subject of the second conjunct (*he*) and attach the parenthetical *I think* as well as *to him* to the subject.

2.1.6 Relative clauses

Several Germanic languages such as German and Dutch show more complex gapping behaviors in sentences with adverbial and relative clauses. For example, Wyngaerd (2007) points out that German also allows a verbal gap in clauses modified by an adverbial clause as in the following example.



'because Peter wanted to visit his friend which reassured me, and Johann (wanted to visit) his children, which amused me'

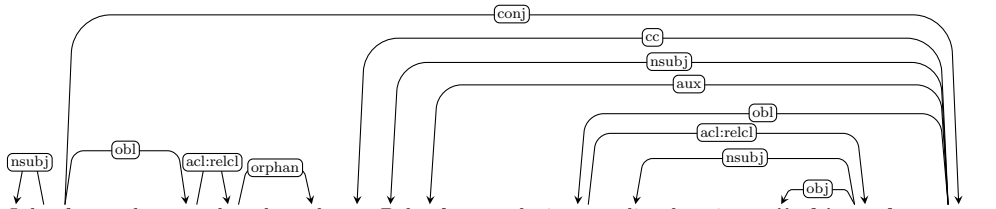
(Wyngaerd 2007)

In this example, the two verbs *besuchen wollte* ('wanted to visit') are missing from the second clause, which leaves three orphans, namely a subject, a direct

7 Source: hhttp://www.ttbook.org/book/transcript/transcript-humour-healing-marc-maron

object, and an adverbial clause without a governor. As in the case of two orphaned constituents, we promote the subject to be the head of the clause as it is the highest type of argument in the obliqueness hierarchy, and we attach the two other constituents to the subject with an `orphan` relation.

Dutch even allows gaps to appear within relative clauses that modify a constituent in each of the conjuncts. In (22), there are in total two elided verbs, and one elided noun. First, the left conjunct is missing the main verb *gesproken* ('talked') in its matrix clause; second, the relative clause of the object in the first conjunct is missing its verb *binnenbracht* ('brought in'); and third, the noun *wijn* ('wine') was elided from the object in the relative clause. The matrix clause of the first conjunct still contains an auxiliary which we promote to be the head of the first conjunct. We further promote the subject of the relative clause, i.e., the relative pronoun, to be the head of the relative clause, and we attach the adjective, which modifies the elided noun, to the promoted subject with an `orphan` relation.



(22) J. heeft met het m. dat de rode en P. heeft met de jongen die de witte **wijn binnenbr. gespr.**
J. has with the girl who the red and P. has with the boy who the white wine in-brought talked

'Jan (talked) to the girl who (brought in) the red (wine), and Piet talked to the boy who brought in the white wine.'

(Wyngaerd 2007)

2.1.7 Discusssion

To the best of my knowledge, these six types of gapping constructions make up all currently documented gapping constructions. As my discussion showed, all of these constructions can be consistently annotated according to the ORPHAN analysis without the need of any additional stipulations. While I omitted the annotations of these sentences according to the COMPOSITE analysis, all of these sentences can also be annotated consistently according to the COMPOSITE proposal. Therefore, with regard to consistency of annotations within and across languages, both analyses seem to be well suited to annotating gapping constructions.

2.2 Dependency structure

The two proposals differ considerably in terms of the resulting dependency structure. The ORPHAN analysis respects conjunct boundaries and treats each conjunct as a syntactic unit, whereas the COMPOSITE analysis leads to a flatter structure in which all orphans are attached directly to the head of the full clause. This raises the question whether there is evidence for the individual conjuncts forming a syntactic unit, which would mean that the ORPHAN analysis leads to a linguistically more reasonable structure than the COMPOSITE analysis.

Many constituency tests such as topicalization, clefting, and stripping suggest that conjuncts with a gap often do not qualify as a constituent. For example, Osborne (2006) argues against treating the gapping in (23) as the coordination of *[the dog a bone]* and *[the man a flower]*, which would suggest that both conjuncts are constituents. He bases his argument on the observation that the former conjunct fails most constituency tests when it is used in a sentence without coordination (24).

(23) She gave the dog a bone, and the man a flower.

(24) a. *The dog a bone, she gave. (Topicalization)
b. *It was the dog a bone that she gave. (Clefting)
c. ?She gave a dog a bone, not a cat some fish. (Stripping)

However, this argument is based on the assumption that *[the dog a bone]* and *[the man a flower]* form a coordinate structure. If we assume instead that the second conjunct is a clause with elided nodes, then none of the above tests seem applicable. At the same time, as already mentioned above, Gerdes & Kahane (2015) point out that phrases such as *"and the man a flower "* can be uttered by a speaker in response to someone else uttering a phrase such as *"she gave the dog a bone"*. They take this behavior as evidence for treating the entire conjunct with a gap (including the conjunction) as a syntactic unit.

Such an analysis is also in line with most accounts of gapping in the generative literature. While there is disagreement on what the deep structure of sentences with gapping should look like and what transformations are employed to derive the surface structure, there is broad consensus that all remnants are part of the same phrase (e.g., Coppock 2001; Johnson 2009). One of the most prominent accounts of gapping in CCG (Steedman 1990) also first combines the phrases within each conjunct and then combines the two conjuncts to form a sentence. I take all of these facts as weak evidence for treating conjuncts with gaps as syntactic units.

Finally, another advantage of the ORPHAN analysis is that this analysis is more consistent with our analysis of other forms of ellipsis. As I mentioned

above, we also promote one word that would depend on the elided word in all other cases of ellipsis, and treating gapping constructions analogously increases the overall consistency of the annotations.

3 Enhanced Representation

As I mentioned earlier, gapping constructions pose challenges for natural language understanding systems as gapped clauses deviate considerably from canonical clause structures and at least the main predicate is missing from the gapped clause. For this and other phenomena, UD also defines an *enhanced* representation, which may be a graph instead of a tree and which may contain additional nodes and relations (Nivre et al. 2016, S. Schuster & Manning 2016). The purpose of this representation is to make implicit relations between content words more explicit in order to facilitate shallow natural language understanding tasks such as relation extraction. One property of the *enhanced* representation is that it resolves gaps by adding nodes to *basic*, i.e., strict surface syntax UD trees. Remnants attach to these additional nodes with meaningful relations just as if nothing had been elided,[8] thus solving the issue of the missing predicate, and in the case of the ORPHAN analysis, the issue of having uninformative `orphan` dependencies. The general idea is to insert as many nodes as required to obtain a structure without orphans while keeping the number of additional nodes to a minimum. On top of additional nodes, we add relations between new nodes and existing content words so that there exist explicit relations between each verb and its arguments and modifiers. Note that this analysis is complementary to the other two proposals. Most automatic parsers do not support inserting copy nodes, so adopting an analysis that includes additional nodes would make UD incompatible with most existing parsers, and therefore we opt for having a strict surface syntax representation as well as a graph-based representation.

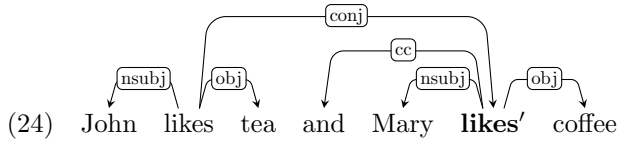
I now illustrate how different cases of gapping can be analyzed in the *enhanced* representation based on several representative examples.

The simplest cases are constructions in which a single verb was elided. In these cases, we insert a copy node[9] of the elided verb at the position of the gap, make this node the head of the conjunct, and attach all orphans to this

8 A similar analysis was used in the tectogrammatical layer of the Prague Dependency Treebank (Bejček et al. 2013).

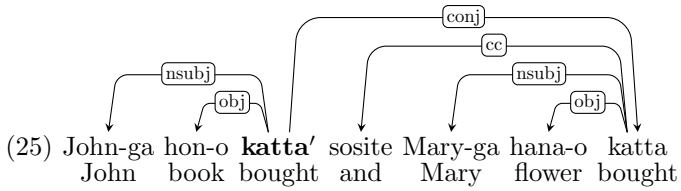
9 Similar copy nodes are already used for some cases of reduced conjunctions in the *collapsed* and *CCprocessed* Stanford Dependencies representations (de Marneffe & Manning 2008) and in the *enhanced* UD representation (S. Schuster & Manning 2016).

copy node. For example, for the following sentence, we insert the copy node *likes′* and attach *Mary* as a subject and *coffee* as an object.



'John likes teas, and Mary coffee.'

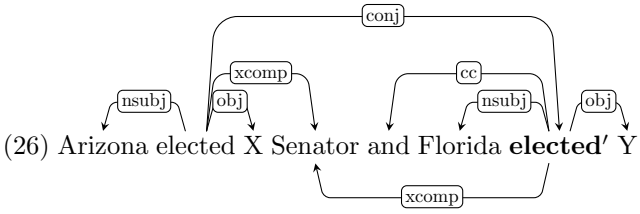
Similarly, we insert a copy node as the new root of a sentence in cases in which the leftmost conjunct contains a gap as, for example, in (25).



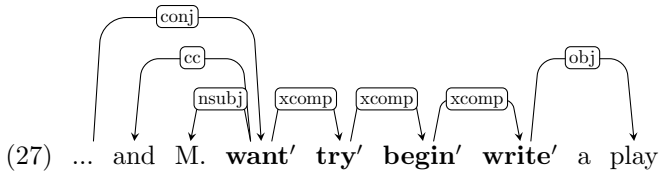
'John bought books, and Mary bought flowers.'

(adapted from Kato (2006))

In cases in which arguments or modifiers were elided along with the verb, we still only insert one copy node for the main verb. However, in order to make the relation between the verb and all of its arguments explicit, we also add relations between the new copy node and existing arguments and meaningful modifiers. In (26), we add a copy node for the elided verb *elected* and a relation between the copy node and *Senator*.



In cases in which a finite verb was elided along with one or more embedded verbs, as in the sentence *"I want to try to begin to write a novel, and Mary a play."*, we insert one copy node for each elided verb. However, unlike in the previous example, we do not add relations between the copy nodes and the semantically vacuous function word *to* because it is not required for the interpretation of the sentence.



The motivation behind these design choices is to have direct and meaningful relations between content words. Many shallow natural language understanding systems, which make use of UD such as open relation extraction systems (Mausam et al. 2012, Angeli et al. 2015) or semantic parsers (Andreas et al. 2016, Reddy et al. 2017), use dependency graph patterns to extract information from sentences. These patterns are typically designed for prototypical clause structures, and by augmenting the dependency graph as described above, many patterns that were designed for canonical clause structures also produce the correct results when applied to sentences with gapping constructions.

4 Parsing Experiments

As I discussed above, there are several theoretical reasons for preferring the ORPHAN analysis over the COMPOSITE analysis of gapping constructions. But as I also mentioned before, we do not want to decide on an analysis exclusively on theoretical grounds but instead, we also want to make sure that we can automatically parse gapping constructions with high accuracy. Based on previous parsing experiments, I expect that the ORPHAN analysis is easier to parse for several reasons. First, it leads to fewer long-distance dependencies as compared to the COMPOSITE analysis and long-distance dependencies are known to be challenging for parsers (McDonald & Nivre 2007). Second, the ORPHAN analysis has only one dependency label for all gapping constructions, whereas the COMPOSITE analysis requires many additional relations, most of which are likely to appear very rarely in the training corpus, which in return makes it challenging for statistical parsers to make meaningful generalizations. That all being said, recently developed parsers such as the graph-based parsers by Kiperwasser & Goldberg (2016) and Dozat & Manning (2017) are much better at dealing with non-projective and long-distance dependencies (Dozat et al. 2017) and therefore some of my assumptions about parseability of the two analyses might no longer hold. I therefore conducted several parsing experiments to determine which of the two proposed analyses is easier to parse. I further investigated two additional related questions. First, is a graph-based parser that takes the entire sentence (and consequently a very large context) into account better at parsing gapping construction than a transition-based parser,

which only takes a very narrow context into account when making parsing decisions? And second, does augmenting the training data with several dozen sentences with gapping constructions significantly improve parsing of sentences with gapping?

4.1 Data

I used the Universal Dependencies English Web Treebank version 2.0 (henceforth EWT; Silveira et al. 2014, Nivre et al. 2017a,b) for training and evaluating parsers. However, as the treebank is relatively small and therefore only contains very few sentences with gapping, I also extracted gapping constructions from the WSJ and Brown portions of the Penn Treebank (Marcus et al. 1993) and the GENIA corpus (Ohta et al. 2002). Further, I copied sentences from the Wikipedia page on gapping constructions[10], from various published papers on gapping constructions, and I made up examples similar to the ones found in the literature. The sentences in the EWT already contain annotations according to the ORPHAN analysis, as well as the copy nodes for the enhanced representation. To obtain the annotations for the sentences from the other three treebanks, I converted them from constituency trees to UD version 1 dependency trees using the Stanford converter (S. Schuster & Manning 2016); then, I converted them to UD version 2 using `udapy` (Popel et al. 2017); and finally, I manually corrected these trees and manually added the empty nodes for the enhanced representation. I used the same conversion pipeline to obtain the dependency trees for the examples from Wikipedia and the linguistic literature but as there exist no gold constituency parses for theses sentences, I first tokenized them using CoreNLP (Manning et al. 2014) and then parsed them to constituency trees using the Stanford Parser (Klein & Manning 2003). These additional trees constitute the GAPPING treebank, which I used in combination with the EWT for some of my parsing experiments. I used the default train/development/test splits of the EWT and I split the GAPPING treebank by putting the first 6 of every 12 sentences in the training split; the 7th, 8th and 9th sentence of every 12 sentences in the development split; and the last 3 of every 12 sentences in the test split, with one exception: My corpus contains four examples similar to (25), each with four varying gap sizes.

(25) I want to try to begin to write a novel, and Mary a play.
(Ross 1970)

In order to make sure that there is no significant lexical overlap and that the parser observes all possible variants of at least one of the examples, I included

10 https://en.wikipedia.org/wiki/Gapping, accessed on Aug 24, 2017

	EWT			GAPPING			EWT + GAPPING (COMBINED)		
	Train	**Dev**	**Test**	**Train**	**Dev**	**Test**	**Train**	**Dev**	**Test**
sentences	12,543	2,002	2,077	83	39	40	12,626	2,041	2,117
tokens	204,585	25,148	25,096	2,105	1,153	1,001	206,690	26,301	26,097
sentences with gapping	19 0.15%	1 0.05%	1 0.05%	83 100%	39 100%	40 100%	102 0.81%	40 1.96%	41 1.94%
copy nodes	21	2	1	113	52	56	134	54	57
unique COMP. relations	18	6	2	32	26	25	39	26	25

Table 1 Treebank statistics. The *copy nodes* row lists the number of copy nodes within each split and the *unique* COMP. *relations* row lists the number of unique composite relations in the treebanks annotated according to the COMPOSITE analysis. The percentages are relative to the total number of sentences.

all variants of two of these examples in the training split, the variants of one in the development split, and the variants of the last one in the test split.

To obtain the treebank with annotations according to the COMPOSITE analysis, I automatically converted the enhanced UD trees. Note that a lossless conversion from the enhanced UD trees is possible using the following procedure. I traverse the dependency tree in depth-first order and whenever I encounter an empty node e, I store the parent relation r_p of e and then I delete e and attach all of its dependents to the governor of e with a relation that is composed of the original parent relation r_p and the relation between e and the dependent.

Table 1 shows several properties of the data splits of the two treebanks as well as their combination.

4.2 Parsers

I used two different dependency parsers, a transition-based parser and a graph-based parser. Both of these parsers use neural-network classifiers to predict the governor and dependency label of each token but they differ in how they construct the dependency tree and how they represent the individual tokens.

Transition-based parser I used the Java implementation of the dependency parser by Chen & Manning (2014). This parser is a transition-based (or shift-reduce) parser, which uses a neural network to predict the next transition. Transition-based parsers consist of a stack, which holds all the processed tokens that have not yet been attached to a governor, and a buffer which contains a few (typically two or three) of the yet unprocessed tokens. These parsers

operate as follows. At each step, they perform one of three actions. First, they can move a token from the buffer to the top of the stack and add a new token to the buffer (SHIFT); second, they can add a dependency arc from the top-most element on the stack to the second element on the stack and remove the second element from the stack (LEFT-ARC); and third, they can add a dependency arc from the second element to the top-most element on the stack and remove the top-most element from the stack (RIGHT-ARC). The parser performs these steps until there are no more tokens on the buffer and there is just one element left on the stack, which will always be the root of the dependency tree. More intuitively, one can imagine the working of the parser as follows. It reads in one token after another and whenever it has read in the governor and the dependent of a relation, it adds an arc between these two tokens. While I described the procedure here using just three possible transitions, in practice, the classifier jointly predicts the direction of an arc and the dependency label, so for a set of L dependency labels, there are $|L|$ LEFT-ARC and another $|L|$ RIGHT-ARC transitions.

Transition-based parsers primarily vary in the type of classifier and the set of features that are used to predict the transitions. The parser by Chen & Manning (2014) embeds tokens, as well as dependency labels and part-of-speech tags into a high-dimensional vector space and then represents a parser state as the concatenation of these embeddings of 1) the three top-most tokens on the stack, 2) some of its children and grand-children, and 3) the three tokens on the buffer. It then passes this input through a cubic activation function which results in a hidden representation that is then passed through a logistic regression classifier in order to predict the most probable transition. The main advantage of this family of parsers is their speed. The number of transitions for a sentence of length n is always exactly $2n - 1$, so it only has to make $O(n)$ predictions.

Note that this parser is greedy, i.e., it makes irreversible parsing decisions before it has encountered all tokens in a sentence, and that it only takes a local context into account, so it bases its decisions for the head of a token only on other tokens in the ultimate linear (tokens on the stack and the buffer) and left structural (tokens on the stack) vicinity. This is the reason why long-distance dependencies are often challenging for this type of parser as it is often hard to get these dependencies right without considering the entire sentence.

Graph-based parser The second parser that I used for my experiments is the dependency parser by Dozat & Manning (2017). This parser is a graph-based parser and differs from the Chen & Manning (2014) parser in terms of how the dependency tree is constructed and how the features are encoded. For

a given sentence, graph-based parsers compute the probability of a dependency $i \rightarrow j$ for each pair of tokens (i, j) s.t. $i \neq j$. This matrix of probabilities can be viewed as the weight of the edges in a fully connected directed graph, in which each vertex corresponds to a token and each edge to a dependency arc. The highest-scoring dependency tree can then be obtained by computing the maximum spanning tree (MST) of this graph, i.e., a tree that contains all the nodes and whose total edge weights are maximal. This results in an unlabeled dependency tree. For each predicted dependency arc, the parser then adds dependency labels in a second step using another classifier.

The parser by Dozat & Manning (2017) uses neural-network classifiers to compute the probabilities of each dependency arc and to determine the labels for each arc. These classifiers extract their features from the hidden states of a bidirectional long short-term memory (biLSTM) network (M. Schuster & Paliwal 1997, Hochreiter & Schmidhuber 1997). biLSTM networks read in a sentence from left to right and from right to left and compute for each token a vector representation that encodes the token itself as well as the preceding and succeeding parts of the entire sentence. Therefore, the representation of each token captures not only the information that is provided by the token itself but also the information that is provided in the sentence which leads to a highly context-specific representation. Apart from making use of more context-sensitive features, this parser also has the advantage that it makes parsing decisions only after it has processed the entire sentence and therefore it is less prone towards making wrong decisions that are caused by partial information.

4.3 Experimental Setup

I used the treebanks with gold tokenization for my experiments. However, I replaced the gold POS tags with predicted fine-grained and universal part-of-speech tags. I used the tagger by Dozat et al. (2017), which I trained on the COMBINED training corpus. Both parsers make use of pre-trained word embeddings and I used the word2vec (Mikolov et al. 2013) embeddings that were provided by the organizers of the CoNLL 2017 Shared Task on Dependency Parsing (Zeman et al. 2017). For training the transition-based parser, I used the default parameters, and for training the graph-based parser, I used the same parameters as Dozat et al. (2017).

I trained each parser on four different training sets: COMBINED-ORPHAN which contains all the training data of the COMBINED dataset annotated according to the ORPHAN analysis; EWT-ORPHAN which contains all the training data of the EWT dataset annotated according to the ORPHAN analysis, and

COMBINED-COMPOSITE and EWT-COMPOSITE, which are the same two datasets annotated according to the COMPOSITE analysis. For the parsers trained on the EWT training data, I used the EWT development set for model selection and for the parsers trained on the COMBINED training data, the COMBINED development set.

4.4 Evaluation

I evaluated the parsers using the two standard metrics for dependency parsing, namely unlabeled attachment score (UAS) and labeled attachment score (LAS). These scores are computed as follows.

$$\text{UAS} = \frac{\text{number of tokens with correct governor}}{\text{number of tokens}} \times 100$$

$$\text{LAS} = \frac{\text{number of tokens with correct governor and dependency label}}{\text{number of tokens}} \times 100$$

I used the official evaluation script of the CoNLL 2007 Shared Task (Nivre et al. 2007) to compute these two scores.

As my primary concern is how well the parsers are able to parse gapping constructions, I further computed the LAS and UAS for all tokens that are governed by a gap (LAS_g and UAS_g).

For pairwise comparisons of results, I use a two-tailed approximate randomization significance test (Yeh 2000, Noreen 1989) to test whether two results differ significantly. I adapted the `sigf` package (Padó 2006) to work with dependency metrics for this purpose.

4.5 Results and Discussion

Tables 2 and 3 show the results of all the parsing models on the development sets and test sets, respectively. Table 4 further shows the results for the relations involved in gapping constructions.

Effect of augmenting the training data Independent of the parser and annotation scheme, augmenting the training data with additional sentences with gapping improved the results on the GAPPING test sets ($p < 0.001$ for the graph-based parser; $p < 0.01$ for the transition-based parser). Further, the variance of the results on the EWT test sets is extremely low (< 0.05) for a given parser and the differences are not statistically significant. This

		EWT		GAPPING		COMBINED	
		LAS	UAS	LAS	UAS	LAS	UAS
Transition-based parser	EWT-ORPHAN	81.38	**85.15**	73.28	79.68	81.03	**84.92**
	COMB.-ORPHAN	**81.43**	85.00	**75.67**	**81.08**	**81.18**	84.83
	EWT-COMPOSITE	81.38	85.06	69.92	75.00	80.88	84.62
	COMB.-COMPOSITE	81.27	84.82	72.41	78.09	80.89	84.53
Graph-based parser	EWT-ORPHAN	87.26	90.52	77.99	84.46	86.86	90.26
	COMB.-ORPHAN	87.29	90.59	**82.77**	**88.75**	**87.09**	**90.51**
	EWT-COMPOSITE	**87.60**	**90.83**	75.10	79.58	87.06	90.34
	COMB.-COMPOSITE	87.25	90.54	81.37	87.44	86.99	90.41

Table 2 Labeled attachment score (LAS) and unlabeled attachment score (UAS) of the various parser models on the three **development** sets. The overall best result in each column is highlighted in **bold** and the best result of the transition-based parser is highlighted in **blue**.

		EWT		GAPPING		COMBINED	
		LAS	UAS	LAS	UAS	LAS	UAS
Transition-based parser	EWT-ORPHAN	80.81	84.26	69.63	75.29	80.39	83.92
	COMB.-ORPHAN	81.10	**84.65**	**74.48**	**80.60**	**80.85**	**84.49**
	EWT-COMPOSITE	80.93	84.42	65.19	71.35	80.33	83.92
	COMB.-COMPOSITE	**81.11**	84.64	69.14	75.29	80.65	84.29
Graph-based parser	EWT-ORPHAN	**87.16**	**90.56**	74.94	80.63	86.70	90.18
	COMB.-ORPHAN	87.13	90.46	**80.51**	**85.61**	**86.88**	**90.27**
	EWT-COMPOSITE	87.14	90.51	72.27	76.68	86.58	89.98
	COMB.-COMPOSITE	87.02	90.32	77.38	83.53	86.65	90.06

Table 3 Labeled attachment score (LAS) and unlabeled attachment score (UAS) of the various parser models on the three **test** sets. The overall best result in each column is highlighted in **bold** and the best result of the transition-based parser is highlighted in **blue**.

		Development		Test	
		LAS_g	UAS_g	LAS_g	UAS_g
Transition-based parser	EWT-ORPHAN	37.89	58.39	31.51	46.58
	COMBINED-ORPHAN	**44.10**	**60.87**	**43.15**	**58.22**
	EWT-COMPOSITE	0.00	16.15	0.00	18.49
	COMBINED-COMPOSITE	4.35	24.22	6.85	25.34
Graph-based parser	EWT-ORPHAN	32.92	52.17	28.08	43.84
	COMBINED-ORPHAN	**60.87**	**76.40**	**57.53**	**67.81**
	EWT-COMPOSITE	0.00	9.94	0.00	7.53
	COMBINED-COMPOSITE	40.37	60.25	33.56	50.68

Table 4 Labeled attachment score (LAS) and unlabeled attachment score (UAS) of the various parser models on the **gapping constructions** in the COMBINED development and test sets.

suggests that adding additional gapping constructions has no significant effect on sentences without gapping but a significant effect on sentences with gapping. The latter result is further supported if we consider the results in Table 4 on only relations involved in gapping constructions. Including more sentences with gapping increased the UAS_g and the LAS_g by up to 51 points. For the graph-based parser, this lead to highly significant improvements in terms of UAS_g and the LAS_g ($p < 0.001$) independent of the annotation scheme; for the transition-based parser, adding more examples significantly increased both scores when parsing to the ORPHAN ANALYSIS ($p < 0.05$) but it did not lead to significant improvements when parsing to the COMPOSITE analysis. All these results suggest that both parsers, and especially the graph-based parser, can make useful generalizations based on the additional gapping constructions present in the training data, but at the same time, that the increased exposure to gapping does not lead the parsers to "hallucinate" gapping constructions.

Effect of the parser Overall, when trained on the same dataset, the graph-based parser performed significantly better than the transition-based parser on all test sets ($p < 0.01$ for all pairwise comparisons of labeled scores; $p < 0.05$ for all comparisons of unlabeled scores). This is not surprising considering that the graph-based parser uses a better token representation and makes global decisions, whereas the transition-based parser takes only the local context into account. Further, the graph-based parser has many more parameters than the transition-based parser, which allows it to learn more fine-grained distinctions than the transition-based parser. Apart from the overall difference in performance, the two parsers also showed differences in how well they were

able to parse gapping constructions to the two different representations, which I discuss in more detail below.

Considering that the two parsers differ along two dimensions, namely their method of constructing the tree as well as their feature representation, it is impossible to distinguish whether the better representation or the non-greedy inference is the main driver of the higher parsing accuracy, but in general it seems that taking all the information of the sentence into account when making parsing decisions is beneficial. Kiperwasser & Goldberg (2016) discuss a transition-based parser that uses a very similar feature representation as the parser by Dozat & Manning (2017), so a future experiment could be to compare a transition-based and a graph-based parser that use a similar feature representation.

Effect of the representation The main question that I tried to answer in my experiments is whether one of the two analyses of gapping is easier to parse. As I mentioned before, I expected the ORPHAN analysis to be easier to parse because of the smaller label space as well as the fewer long-distance dependencies. This seems to be true for the transition-based parser. The transition-based parser performed significantly better when trained on data that was annotated according to the ORPHAN analysis independent of the training set (LAS differences: $p < 0.05$; UAS differences: $p < 0.01$). If one only considers the relations involved in gapping constructions in Table 4, it seems as if my predictions were right. Both the UAS_g and the LAS_g on the test set are extremely low for the model that was trained on the COMBINED COMPOSITE training data. The low LAS_g suggests that the model is struggling with choosing the right relation from the large label space, and the low UAS_g suggests that the model is struggling with the attachment, presumably due to its inability to reconstruct the numerous long-distance dependencies. The model that was trained on the COMBINED ORPHAN data, on the other hand, is significantly better ($p < 0.001$) at choosing the correct attachment and dependency label as compared to the model trained on the COMBINED COMPOSITE data.

For the graph-based parser, these findings also hold but the differences are smaller. On the GAPPING test set, the graph-based parser performed significantly better ($p < 0.001$) when trained on the ORPHAN data as compared to the COMPOSITE data. However, as shown in Table 4, the difference between representations in terms of UAS_g and the LAS_g is much smaller (but still significant, $p < 0.01$) for the graph-based parser. The larger number of parameters of this parser, seems to allow it to learn better distinctions between the various composite relation labels and therefore this parser seems to struggle less with the large label space. However, if we consider the models that were trained on

the EWT, which only contains a small number of sentences with gapping, we observe a stark contrast between the two representations. Not surprisingly, if the parser only observed very few different composite labels in the training data, it fails to assign the correct label to many constructions in the test data because it can only output relations that were present in the training data. Therefore the LAS_g on the test set of the EWT COMPOSITE model is much lower (0.00) than the score of the COMBINED COMPOSITE model (33.56), which was trained on many different gapping constructions. For the models trained on the ORPHAN representation, fewer training examples are less of an issue. In fact, there is no significant difference in terms of LAS_g and UAS_g between the COMBINED COMPOSITE and the EWT ORPHAN model despite the fact that the latter was trained on much fewer gapping constructions. While augmenting the training data with more examples also helped performance of the ORPHAN analysis parser ($p < 0.001$), this suggests that the ORPHAN analysis makes it much easier for parsers to learn useful generalizations from a realistic number of examples while the COMPOSITE analysis requires many more training examples.

In conclusion, while the graph-based parser is able to parse some gapping constructions to the COMPOSITE representation when it was trained on a sufficient number of examples, the results of these experiments suggest that the ORPHAN representation is easier to parse. This is in particular true for the transition-based parser, but to a large extent also for the graph-based parser.

5 Enhancement experiments

Lastly, I try to answer the question of which of the two analyses leads to a representation that is better suited to downstream natural language understanding tasks. As I mentioned before, many downstream systems use dependency patterns that are designed for canonical clause structures and therefore it is reasonable to assume that neither of the two proposals are good representations for downstream tasks. However, as the enhanced representation mitigates the problem of non-conventional clause structures to a large degree, we can expect this representation to be well suited for downstream tasks. Extrinsic evaluations are very time-consuming because one has to train many complex downstream systems. Further, given the fact that gapping constructions are relatively rare, it is unlikely that one would see statistically significant differences if one evaluated different representations on existing downstream tasks. I therefore approximate the degree of usefulness in downstream tasks with how well one can automatically reconstruct the enhanced representation from surface syntax dependency trees, instead of actually performing an extrinsic evaluation.

The two analyses of gapping contain very different information and therefore require different procedures to derive the enhanced representation.

5.1 Orphan procedure

The ORPHAN analysis of gapping constructions provides information on which arguments are dependents of an elided predicate. However, it lacks information on the types of arguments as well as on what exactly has been elided. The task of obtaining the enhanced representation therefore consists of determining and copying the tokens that have been elided as well as determining to which of the copied tokens using which relation the orphaned arguments should be attached.

In developing a procedure to perform this task, I made use of the fact that in the vast majority of cases, all the types arguments and modifiers that are expressed in the conjunct with a gap are also expressed in the full conjunct. The problem of determining which nodes to copy and which relations to use can therefore be reduced to the problem of aligning arguments in the gapped conjunct to the arguments in the full conjunct. I devised the following procedure to obtain the enhanced representation from a tree that was annotated according to the ORPHAN analysis.

i. Create a list F of arguments of the head of the full conjunct by considering all core argument dependents of the conjunct's head as well as clausal and nominal non-core dependents.

ii. Create a list G of arguments in the gapped conjunct by considering the head of the gapped conjunct as well as all its `orphan` dependents.

iii. Find the highest-scoring alignment of the arguments in G to the arguments in F.

iv. Copy the head of the full conjunct and attach the copy node c to the head of the full conjunct.

v. For each node g_i that has been aligned to f_j, attach g_i to c with the same relation as the parent relation of f_j, e.g., if f_j is attached to the head of the full conjunct with an `nsubj` relation, also attach g_i to c with an `nsubj` relation.

vi. Repeat steps i.-v. as long as there are remaining conjuncts with gaps.

A crucial step in this procedure is the third step, determining the highest-scoring alignment. This can be done straightforwardly with the sequence

alignment algorithm by Needleman & Wunsch (1970) if one defines a similarity function $sim(g_i, f_j)$ that returns a similarity score between the arguments g_i and f_j. I defined sim based on the intuitions that often, parallel arguments are of the same syntactic category, that they are introduced by the same function words (e.g., the same preposition), and that they are closely related in meaning. The first intuition can be captured by checking whether the part-of-speech tags of the heads of the two arguments match. The other two intuitions can be captured by embedding each of the two arguments into a high-dimensional vector-space and then computing their euclidean distance. In order to embed the arguments, I retrieve the vector for each of the tokens in each of the arguments from a pre-trained 100-dimensional GloVe (Pennington et al. 2014) word embedding matrix and then average the token vectors to obtain an argument vector. Given the POS tags t_{g_i} and t_{f_j} and the argument embeddings v_{g_i} and v_{f_j}, sim is defined as follows.

$$sim(g_i, f_j) = -\|v_{g_i} - v_{f_j}\|_2 + \mathbb{1}\left[t_{g_i} = t_{f_j}\right] \times pos\_mismatch\_penalty$$

$pos\_mismatch\_penalty$ is an empirical parameter which I set to -2 as this value led to the best performance on the training data.

This procedure can be used as described for almost all sentences with gapping constructions. However, if parts of an argument were elided along the main predicate, it can become necessary to copy more than one node. I solved this issue by considering the alignment not only between complete arguments in the full clause and the gapped clause but also between partial arguments in the full clause and the complete arguments in the gapped clause. For example, for the sentence "*Mary wants to write a play and Sue a book*" the complete arguments of the full clause are {*Mary*, *to write a play*} and the arguments of the gapped clause are {*Mary*, *a book*}. In this case, I also consider the list of partial arguments {*Mary*, *a play*} and if the arguments of the gapped conjunct align better to the list of partial arguments, I use this alignment. However, now that the token *write* is part of the dependency path between *want* and *play*, I also have to make a copy of *write* to reconstruct the enhanced representation of the gapped clause.

5.2 Composite procedure

The COMPOSITE analysis of gapping constructions provides information on which arguments are dependents of an elided predicate, as well as the type of argument. It has therefore been argued that one can deterministically obtain the enhanced representation from trees annotated with composite relations.

For most sentences, this is indeed true and it is possible to obtain the enhanced representation using the following procedure.

i. Go through each edge $e = (gov, dep, reln)$ of the dependency tree.

ii. If $reln$ is a composite relation, perform the following steps.

   a. Split $reln$ into its two atomic parts r_1 and r_2.

   b. Remove the original edge e from the dependency tree.

   c. If no copy of gov exists, make a copy gov' of gov and attach gov' to gov with the relation r_1.

   d. Attach dep to gov' with the relation r_2.

For simplicity, I only outlined here the procedure to obtain the enhanced representation from dependency trees with relations that are composed of at most two atomic relations. However, as previously mentioned, composite relations that consist of more than two relations such as `conj>xcomp>xcomp>obj` are also possible. The described procedure can be easily turned into a recursive procedure that can also deal with these relations, which I did for my actual implementation.

One complication of this procedure is that it cannot properly deal with multiple conjuncts with gapping. For a sentence such as "*Mary drinks tea, Sue coffee, and Paul orange juice*", the enhanced dependency graph should contain two copy nodes of the predicate *drinks*. However, as all the orphans are attached directly to the head of the first conjunct in the COMPOSITE analysis, the dependency tree lacks a marking of conjunct boundaries and hence, it is impossible to determine the number of required copy nodes and which orphans should be attached to which of the copies if one only considers the tree structure. While it is conceivable to use heuristics (e.g., by looking for commas or coordinating conjunctions) to determine conjunct boundaries, this defeats the idea of using a dependency tree analysis that allows one to deterministically obtain the enhanced representation, and therefore I did not implement any such heuristics.

5.3 Experiments

I conducted two experiments. First, I conducted an *oracle* experiment in which I evaluated how well one can reconstruct the enhanced dependency graph from gold dependency trees annotated according to the two different analyses using the procedures that I described in the previous sections. This experiment was

intended to answer the question which of the two procedures works better if we assume that we have a parser that can perfectly parse to either of the two representations.

Second, I evaluated how well one can reconstruct the enhanced dependency graphs from dependency trees that were output by a parser. This *in-vivo* experiment was intended to answer the question which of the two analyses leads to better enhanced graphs in a realistic end-to-end setting. For this experiment, I used the output of the graph-based parser trained on the COMBINED training set.

5.3.1 Evaluation

I evaluated the output of the two procedures on the COMBINED development and test sets. Considering that the outputs are graphs rather than trees, I used the labeled and unlabeled precision and recall metrics, which are generally used to evaluate dependency graphs (e.g., Oepen et al. 2015). However, considering that my procedures only change edges that are involved in gapping constructions and considering that these edges only make up a very small fraction of all the edges in the treebank, I computed these metrics only on edges that are involved in gapping constructions, i.e., only incoming and outgoing edges of copy nodes. I further excluded edges going to punctuation and coordinating conjunctions as they are less relevant for most practical tasks and including these edges could potentially lead to inflated scores. Unlabeled precision (UP) and recall (UR) are defined as follows.

$$UP = \frac{\text{number of edges with correct governor and dependent}}{\text{number of edges in predicted graph}} \times 100$$

$$UR = \frac{\text{number of edges with correct governor and dependent}}{\text{number of edges in gold graph}} \times 100$$

The labeled variants of these two metrics are the same with the exception that an edge is only counted as correct if its governor, dependent, and label are correct.

5.4 Results and Discussion

Table 5 shows the results for the two experiments on the COMBINED development and test sets.

Oracle experiment The top two rows of Table 5 show the results for the oracle experiment. Considering that there are only around 40 sentences with

		Development				Test			
		LP	LR	UP	UR	LP	LR	UP	UR
Gold graphs	ORPHAN	**87.65**	**88.20**	**93.21**	**93.79**	83.77	85.06	90.57	**93.51**
	COMP.	86.54	83.85	86.54	83.85	**91.72**	**93.51**	**91.72**	**93.51**
Pred. graphs	ORPHAN	**85.48**	32.92	**88.71**	34.16	**86.44**	**33.12**	**88.14**	33.77
	COMP.	60.64	**35.40**	64.89	**37.89**	69.01	31.82	74.65	**34.42**

Table 5 Labeled precision (LP) and recall (LR) and unlabeled precision (UP) and recall (UR) of the two enhancement procedures on the **gapping constructions** in the COMBINED development and test sets. The top two rows show the results for enhancing gold dependency trees; the bottom two rows show the results for enhancing predicted dependency trees.

around 200 edges that are part of gapping constructions in each of the two data sets, the differences between the two methods are quite small. Still, on the test set, the procedure that reconstructs the enhanced graph from trees with the COMPOSITE relations performed significantly better ($p < 0.05$) in terms of the labeled metrics than the other procedure, but there was no significant difference between the performance of the two procedures in terms of the unlabeled precision and recall. Note that, as expected, the labeled and unlabeled scores are identical for the COMPOSITE procedure. This procedure only has to determine which nodes to copy and where orphaned arguments should be attached but as the correct label is included in the BASIC representation, this procedure always assigns the correct label if the composite label is correct.

Not surprisingly, these two procedures struggle with different constructions. As mentioned before, the COMPOSITE procedure cannot properly reconstruct enhanced graphs of sentences with multiple gapped conjuncts and these sentences are responsible for almost all of the remaining errors. Further, there is the very rare case in which the main predicate has multiple modifiers with the same relation, e.g., multiple adverbial clause modifiers. If this is the case, a composite relation such as `conj>advcl>obj` does not uniquely identify the argument in the full conjunct and one might end up copying the wrong predicate.

The main source of error for the ORPHAN procedure is sentences in which the gapped conjunct contains arguments or modifiers without a correspondent in the the full conjunct. For example, in the sentence "*They left the company, many for good.*", the oblique modifier *for good* does not have a correspondent, and therefore, it cannot be aligned to any argument in the first conjunct. In such cases, the ORPHAN procedure cannot determine the correct label and assigns

the most general `dep` label. Lastly, in some cases, the ORPHAN procedure also copies nodes for other types of ellipsis. For example, the enhanced graph of the sentence "*Delivery of the first aircraft is set for early November and a second for December*" as output by the ORPHAN procedure contains copy nodes of the main predicate *set* as well as the elided head noun *delivery*. However, according to the UD guidelines, which are reasonably designed to keep the number of copy nodes to a minimum, the elided head noun should not be copied, so this is marked as incorrect despite actually being potentially useful for downstream applications.

Overall, these numbers suggest that both procedures can reconstruct the enhanced graphs from perfectly annotated trees with high accuracy. While this was expected for the COMPOSITE procedure, this is more surprising for the ORPHAN procedure which not only has to reconstruct the structure but also the labels and still performs almost on par with the COMPOSITE procedure.

In-vivo experiments The purpose of the in-vivo experiment was to evaluate which one of the two procedures works better in practice. As my parsing experiments showed, even state-of-the-art parsers still make many mistakes in parsing gapping constructions, independent of which representation was used. This fact is also reflected in the numbers in the bottom two rows of Table 5, which are much lower than the oracle numbers. Recall is very low for both procedures as both of them rely on the parser to detect gapping constructions. If the parser successfully detected a gapping construction, the ORPHAN procedure seems to be better at reconstructing the enhanced graph and assigning the labels, which is reflected in the higher labeled and unlabeled precision values. While none of these differences were significant, this still suggests that the parser that was trained to parse to the COMPOSITE analysis struggles with assigning the correct composite label, which then translates to lower precision of the COMPOSITE procedure.

The numbers in Table 5 suggest that overall, parsing to the ORPHAN analysis and then reconstructing the enhanced graphs from this representation works at least as well (no statistically significant differences in terms of any of the four metrics) as first parsing to the COMPOSITE analysis. However, with both procedures, recall is still very low and both of them still miss almost two thirds of the edges involved in gapping in the two test sets.

6 Conclusion

In this paper, I discussed two different analyses of gapping constructions within the Universal Dependencies framework. I primarily evaluated the two analyses

along three dimensions, namely theoretical considerations, their parseability, and how easy they make it to reconstruct the enhanced representation. I argued that from a theoretical point of view, the ORPHAN analysis is preferable as it respects clause boundaries and leads to more theory-internal consistency. The results of the parsing and enhancement experiments suggest that the ORPHAN analysis is easier to parse and is on par with the COMPOSITE analysis in terms of reconstruction of the enhanced graphs. These three results, in combination with the arguably easier annotation, make the ORPHAN analysis the preferred annotation according to the Universal Dependencies design goals.

From a more practical point of view, my parsing experiments have shown that while the new generation of parsers that was developed over the past two years is much better at parsing gapping constructions, we are still quite far away from reliably parsing these constructions. At the same time, as I demonstrated in my enhancement experiments, it is possible to reconstruct the enhanced dependency graphs from correct dependency trees with very high accuracy, so as parsers become more and more accurate, we can also expect to get closer to reliably parsing sentences with gapping to a representation that is useful for downstream tasks.

References

Andreas, Jacob, Marcus Rohrbach, Trevor Darrell & Dan Klein. 2016. Learning to compose neural networks for question answering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2016)*.

Angeli, Gabor, Melvin Johnson Premkumar & Christopher D Manning. 2015. Leveraging linguistic structure for open domain information extraction. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*.

Bejček, Eduard, Eva Hajičová, Jan Hajič, Pavlína Jínová, Václava Kettnerová, Veronika Kolářová, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Jarmila Panevová, Lucie Poláková, Magda Ševčíková, Jan Štěpánek & Šárka Zikánová. 2013. *Prague Dependency Treebank 3.0*. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University. http://hdl.handle.net/11858/00-097C-0000-0023-1AAF-3.

Bozsahin, Cem. 2000. Gapping and word order in Turkish. In *Proceedings of the 10th International Conference on Turkish Linguistics*, 58–66.

Chen, Danqi & Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 740–750.

Coppock, Elizabeth. 2001. Gapping: In defense of deletion. In Mary Andronis, Christopher Ball, Heidi Elston & Sylvain Neuvel (eds.), *Papers from the 37th Meeting of the Chicago Linguistic Society*, 133–148. Chicago Linguistic Society.

de Marneffe, Marie-Catherine, Bill MacCartney & Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*, 449–454.

de Marneffe, Marie-Catherine & Christopher D. Manning. 2008. *Stanford typed dependencies manual*. Tech. rep.

Dozat, Timothy & Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*, 1–8.

Dozat, Timothy, Peng Qi & Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 Shared Task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, 20–30.

Farudi, Annahita. 2013. *Gapping in Farsi: A Crosslinguistic Investigation*. MIT PhD thesis.

Gerdes, Kim & Sylvain Kahane. 2015. Non-constituent coordination and other coordinative constructions as dependency graphs. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, 101–110.

Hochreiter, Sepp & Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8). 1735–1780.

Jackendoff, Ray S. 1971. Gapping and related rules. *Linguistic Inquiry* 2(1). 21–35.

Johnson, Kyle. 2009. Gapping is not (VP-) ellipsis. *Linguistic Inquiry* 40(2). 289–328.

Kato, Kumiko. 2006. *Japanese Gapping in Minimalist Syntax*. University of Washington PhD thesis.

Kiperwasser, Eliyahu & Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics* 4. 313–327.

Klein, Dan & Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL 2003)*, 423–430.

Kush, Dave. 2016. Notes on gapping in Hindi-Urdu: Conjunct size and focus parallelism. *Linguistic Analysis* 40(3-4). 255–296.

Manning, Christopher D., John Bauer, Jenny Finkel, Steven J Bethard, Mihai Surdeanu & David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 55–60.

Marcus, Mitchell P, Beatrice Santorini & Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2). 313–330.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland & Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

McDonald, Ryan & Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, 122–131.

Mel'čuk, Igor A. 1988. *Dependency Syntax: Theory and Practice*. Albany: State University of New York Press.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado & Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, 3111–3119.

Needleman, Saul B. & Christian D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology* 48(3). 443–453.

Nivre, Joakim, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel & Deniz Yuret. 2007. The CoNLL 2007 Shared Task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*.

Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty & Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, 1659–1666.

Nivre, Joakim et al. 2017a. *Universal Dependencies 2.0*. http://hdl.handle.net/11234/1-1983.

Nivre, Joakim et al. 2017b. *Universal Dependencies 2.0 – CoNLL 2017 Shared Task development and test data*. http://hdl.handle.net/11234/1-2184.

Noreen, Eric W. 1989. *Computer-Intensive Methods for Testing Hypotheses*. New York, NY: John Wiley & Sons.

Oepen, Stephan, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic & Zdenka Uresova. 2015. SemEval 2015 Task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, 915–926.

Ohta, Tomoko, Yuka Tateisi & Jin-Dong Kim. 2002. The GENIA corpus: An annotated research abstract corpus in molecular biology domain. *Proceedings of the Second International Conference on Human Language Technology Research*. 82–86.

Osborne, Timothy. 2006. Shared material and grammar: Toward a dependency grammar theory of non-gapping coordination for English and German. *Zeitschrift für Sprachwissenschaft* 25(1).

Padó, Sebastian. 2006. *User's guide to* `sigf`*: Significance testing by approximate randomisation*. https://nlpado.de/~sebastian/software/sigf.shtml.

Pennington, Jeffrey, Richard Socher & Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 1532–1543.

Pollard, Carl & Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Popel, Martin, Zdeněk Žabokrtský & Martin Vojtek. 2017. Udapi: Universal API for Universal Dependencies. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, 96–101.

Reddy, Siva, Oscar Täckström, Slav Petrov, Mark Steedman & Mirella Lapata. 2017. Universal semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, 89–101.

Ross, John R. 1967. *Constraints on Variables in Syntax*. MIT PhD thesis.

Ross, John R. 1970. Gapping and the order of constituents. In Manfred Bierwisch & Karl Erich Heidolph (eds.), *Progress in Linguistics*, 249–259. The Hague: De Gruyter.

Schuster, Mike & Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11). 2673–2681.

Schuster, Sebastian & Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, 2371–2378.

Silveira, Natalia, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer & Christopher D Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014)*.

Steedman, Mark. 1990. Gapping as constituent coordination. *Linguistics and Philosophy* 13(2). 207–263.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez & Joakim Nivre. 2008. The CoNLL-2008 Shared Task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL 08)*, 159–177.

Wyngaerd, G. Vanden. 2007. Gapping constituents. unpublished manuscript, FWO/K.U. Brussel. https://lirias.kuleuven.be/bitstream/123456789/408979/1/09HRPL&L02.pdf.

Yeh, Alexander. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the The 18th International Conference on Computational Linguistics (COLING 2000)*, 947–953.

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 Gökırmak, Anna Nedoluzhko, Silvie Cinková, Jan Hajič jr., Jaroslava Hlaváčová, Václava Kettnerová, Zdenka 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, Hěctor Martínez Alonso, 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 Strnadova, 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 *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, 1–19.