Sentences with Gapping: Parsing and Reconstructing Elided Predicates

Sebastian Schuster **Joakim Nivre**[‡] **Christopher D. Manning**

Departments of Linguistics and Computer Science, Stanford University
{sebschu,manning}@stanford.edu
[‡]Department of Linguistics and Philology, Uppsala University
joakim.nivre@lingfil.uu.se

Abstract

Sentences with gapping, such as *Paul likes coffee and Mary tea*, lack an overt predicate to indicate the relation between two or more arguments. Surface syntax representations of such sentences are often produced poorly by parsers, and even if correct, not well suited to downstream natural language understanding tasks such as relation extraction that are typically designed to extract information from sentences with canonical clause structure. In this paper, we present two methods for parsing to a Universal Dependencies graph representation that explicitly encodes the elided material with additional nodes and edges. We find that both methods can reconstruct elided material from dependency trees with high accuracy when the parser correctly predicts the existence of a gap. We further demonstrate that one of our methods can be applied to other languages based on a case study on Swedish.

1 Introduction

Sentences with gapping (Ross, 1970) such as *Paul likes coffee and Mary tea* are characterized by having one or more conjuncts that contain multiple arguments or modifiers of an elided predicate. In this example, the predicate *likes* is elided for the relation *Mary likes tea*. While these sentences appear relatively infrequently in most written texts, they are often used to convey a lot of factual information that is highly relevant for language understanding (NLU) tasks such as open information extraction and semantic parsing. For example, consider the following sentence from the WSJ portion of the Penn Treebank (Marcus et al., 1993).

(1) Unemployment has reached 27.6% in Azerbaijan, 25.7% in Tadzhikistan, 22.8% in Uzbekistan, 18.8% in Turkmenia, 18% in Armenia and 16.3% in Kirgizia, [...]

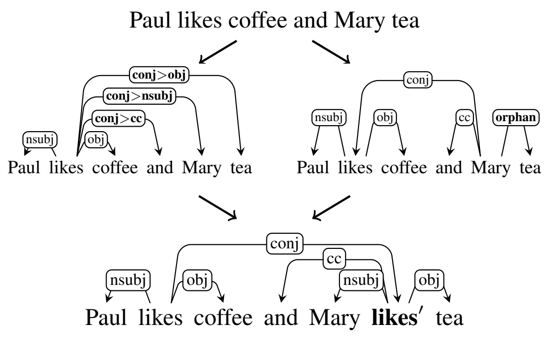

Paul likes coffee and Mary tea
conj>obj
conj>nsubj
conj>cc
nsubj
obj
Paul likes coffee and Mary tea
conj
nsubj
obj
cc
orphan
Paul likes coffee and Mary tea
conj
cc
nsubj
obj
nsubj
obj
Paul likes coffee and Mary **likes**′ tea

Figure 1: Overview of our two approaches. Both methods first parse a sentence with gapping to one of two different dependency tree representations and then reconstruct the elided predicate from this tree.

To extract the information about unemployment rates in the various countries, an NLU system has to identify that the percentages indicate unemployment rates and the locational modifiers indicate the corresponding country. Given only this sentence, or this sentence and a strict surface syntax representation that does not indicate elided predicates, this is a challenging task. However, given a dependency graph that reconstructs the elided predicate for each conjunct, the problem becomes much easier and methods developed to extract information from dependency trees of clauses with canonical structures are much more likely to extract the correct information from a gapped clause.

While gapping constructions receive a lot of attention in the theoretical syntax literature (e.g., Ross 1970; Jackendoff 1971; Steedman 1990; Coppock 2001; Osborne 2006; Johnson 2014; Toosarvandani 2016; Kubota and Levine 2016), they have been almost entirely neglected by the NLP community so far. The Penn Treebank explicitly annotates gapping constructions, by co-indexing arguments in the clause with a predicate and the clause with the gap, but these co-indices are not included in the standard parsing metrics and almost all parsers ignore them.[1] Despite the sophisticated analysis of gapping within CCG (Steedman, 1990), sentences with gapping were deemed too difficult to represent within the CCG-Bank (Hockenmaier and Steedman, 2007). Similarly the treebanks for the Semantic Dependencies Shared Task (Oepen et al., 2015) exclude all sentences from the Wall Street Journal that contain gapping. Finally, while the tectogrammatical layer of the Prague Dependency Treebank (Bejček et al., 2013) as well as the enhanced Universal Dependencies (UD) representation (Nivre et al., 2016) provide an analysis with reconstructed nodes for gapping constructions, there exist no methods to automatically parse to these representations.

Here, we provide the first careful analysis of parsing of gapping constructions, and we present two methods for reconstructing elided predicates in sentences with gapping within the UD framework. As illustrated in Figure 1, we first parse to a dependency tree and then reconstruct the elided material. The methods differ in how much information is encoded in the dependency tree. The first method adapts an existing procedure for parsing sentences with elided function words (Seeker et al., 2012), which uses composite labels that can be deterministically turned into dependency graphs in most cases. The second method is a novel procedure that relies on the parser only to identify a gap, and then employs an unsupervised method to reconstruct the elided predicates and reattach the arguments to the reconstructed predicate. We find that both methods can reconstruct elided predicates with very high accuracy from gold standard dependency trees. When applied to the output of a parser, which often fails to identify gapping, our methods achieve a sentence-level accuracy of 32% and 34%, significantly outperforming the recently proposed constituent parser by Kummerfeld and Klein (2017).

2 Background

2.1 Gapping constructions

Gapping constructions in English come in many forms that can be broadly classified as follows.

(2) *Single predicate gaps*:
John **bought** books, and Mary ___ flowers.

(3) *Contiguous predicate-argument gap (including ACCs)*:
Eve **gave flowers** to Al and Sue ___ to Paul.
Eve **gave** a CD to Al and ___ roses to Sue.

(4) *Non-contiguous predicate-argument gap*:
Arizona **elected** Goldwater **Senator**, and Pennsylvania ___ Schwelker ___.
(Jackendoff, 1971)

(5) *Verb cluster gap*:
I **want to try to begin to write a novel** and
... Mary ___ a play.
... Mary ___ to write a play.
... Mary ___ to begin to write a play.
... Mary ___ to try to begin to write a play.
(Ross, 1970)

The defining characteristic of gapping constructions is that there is a clause that lacks a predicate (the *gap*) but still contains two or more arguments or modifiers of the elided predicate (the *remnants* or *orphans*). In most cases, the remnants have a corresponding argument or modifier (the *correspondent*) in the clause with the overt predicate.

These types of gapping also make up the majority of attested constructions in other languages. However, Wyngaerd (2007) notes that Dutch permits gaps in relative clauses, and Farudi (2013) notes that Farsi permits gaps in finite embedded clauses even if the overt predicate is not embedded.[2]

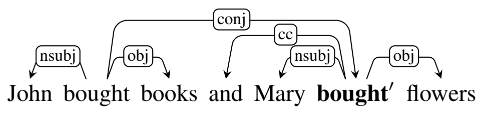
2.2 Target representation

We work within the UD framework, which aims to provide cross-linguistically consistent dependency annotations that are useful for NLP tasks. UD defines two types of representation: the *basic* UD representation which is a strict surface syntax dependency tree and the *enhanced* UD representation (Schuster and Manning, 2016) which may be a graph instead of a tree and may contain additional nodes. The analysis of gapping in the enhanced representation makes use of copy nodes for elided predicates and additional edges for elided arguments, which we both try to automatically reconstruct in this paper. In the simple case in which only one predicate was elided, there is exactly one

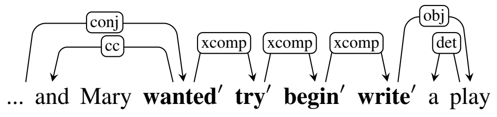
[1] To the best of our knowledge, the parser by Kummerfeld and Klein (2017) is the only parser that tries to output the co-indexing of constituents in clauses with gapping but they lack an explicit evaluation of their co-indexing prediction accuracy.

[2] See Johnson (2014) or Schuster et al. (2017) for a more comprehensive overview of cross-linguistically attested gapping constructions.

copy node for the elided predicate, which leads to a structure that is identical to the structure of the same sentence without a gap.[3]

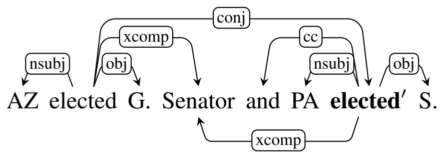


If a clause contains a more complex gap, the enhanced representation contains copies for all content words that are required to attach the remnants.


conj
cc
xcomp
xcomp
xcomp
obj
det
... and Mary **wanted**′ **try**′ **begin**′ **write**′ a play

The motivation behind this analysis is that the semantically empty markers *to* are not needed for interpreting the sentence and minimizing the number of copy nodes leads to less complex graphs.

Finally, if a core argument was elided along with the predicate, we introduce additional dependencies between the copy nodes and the shared arguments, as for example, the open clausal complement (xcomp) dependency between the copy node and *Senator* in the following example.

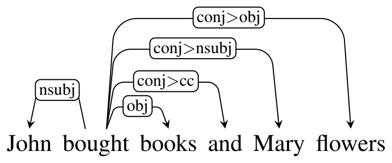


The rationale for not copying all arguments is again to keep the graph simple, while still encoding all relations between content words. Arguments can be arbitrarily complex and it seems misguided to copy entire subtrees of arguments which, e.g., could contain multiple adverbial clauses. Note that linking to existing nodes would not work in the case of verb clusters because they do not satisfy the subtree constraint.

3 Methods

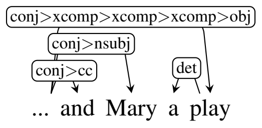
3.1 Composite relations

Our first method adapts one of the procedures by Seeker et al. (2012), which represents gaps in dependency trees by attaching dependents of an elided predicate with composite relations. These relations represent the dependency path that would have existed if nothing had been elided. For example, in the following sentence, the verb *bought*, which would have been attached to the head of the first conjunct with a conj relation, was elided from the second conjunct and hence all nodes that would have depended on the elided verb, are attached to the first conjunct using a composite relation consisting of conj and the type of argument.



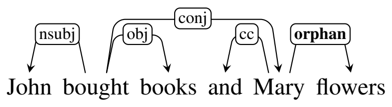
The major advantage of this approach is that the dependency tree contains information about the types of arguments and so it should be straightforward to turn dependency trees of this form into enhanced UD graphs. For most dependency trees, one can obtain the enhanced UD graph by splitting the composite relations into its atomic parts and inserting copy nodes at the splitting points.[4]

At the same time, this approach comes with the drawback of drastically increasing the label space. For sentences with more complex gaps as in (5), one has to use composite relations that consist of more than two atomic relations and theoretically, the number of composite relations is unbounded:



3.2 Orphan procedure

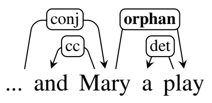
Our second method also uses a two-step approach to resolve gaps, but compared to the previous method, it puts less work on the parser. We first parse sentences to the basic UD v2 representation, which analyzes gapping constructions as follows. One remnant is promoted to be the head of the clause and all other remnants are attached to the promoted phrase. For example, in this sentence, the subject of the second clause, *Mary*, is the head of the clause and the other remnant, *flowers*, is attached to *Mary* with the special orphan relation:



[3]To enhance the readability of our examples, we place the copy node in the sentence where the elided predicate would have been pronounced. However, as linear order typically does not matter for extracting information with dependency patterns, our procedures only try to recover the structure of canonical sentences but not their linear order.

[4]Note that this representation does not indicate conjunct boundaries, and for sentences with multiple gapped conjuncts, it is thus unclear how many copy nodes are required.

This analysis can also be used for more complex gaps, as in the example with a gap that consists of a chain of non-finite embedded verbs in (5).



When parsing to this representation, the parser only has to identify that there is a gap but does not have to recover the elided material or determine the type of remnants. As a second step, we use an unsupervised procedure to determine which nodes to copy and how and where to attach the remnants. In developing this procedure, we made use of the fact that in the vast majority of cases, all arguments and modifiers that are expressed in gapped conjunct are also expressed in the full conjunct. The problem of determining which nodes to copy and which relations to use can thus be reduced to the problem of aligning arguments in the gapped conjunct to arguments in the full conjunct. We apply the following procedure to all sentences that contain at least one `orphan` relation.

1. 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, and adverbial modifiers.
2. Create a list G of arguments in the gapped conjunct that contains the head of the gapped conjunct and all its `orphan` dependents.
3. Find the highest-scoring monotonic alignment of arguments in G to arguments in F.
4. Copy the head of the full conjunct and attach the copy node c to the head of the full conjunct with the original relation of the head of the gapped conjunct (usually `conj`).
5. For each argument $g \in G$ that has been aligned to $f \in F$, attach g to c with the same relation as the parent relation of f, e.g., if f is attached to the head of the full conjunct with an `nsubj` relation, also attach g to c with an `nsubj` relation. Attach arguments $g' \in G$ that were not aligned to any token in F to c using the general `dep` relation.
6. For each copy node c, add dependencies to all core arguments of the original node which do not have a corresponding remnant in the gapped clause. For example, if the full conjunct contains a subject, an object, and an oblique modifier but the clause with the gap, only a subject and an oblique modifier, add an object dependency between the copy node and the object in the full conjunct.

A crucial step is the third step, determining the highest-scoring alignment. This can be done straightforwardly with the sequence alignment algorithm by Needleman and Wunsch (1970) if one defines a similarity function $sim(g, f)$ that returns a similarity score between the arguments g and f. We 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 penalizing mismatching POS tags, and the other two by computing the distance between argument embeddings. We compute these embeddings by averaging over the 100-dim. pretrained GloVe (Pennington et al., 2014) embeddings for each token in the argument. Given the POS tags t_g and t_f and the argument embeddings v_g and v_f, sim is defined as follows.[5]

$$sim(g, f) = -\|v_g - v_f\|_2 + \mathbb{1}\left[t_g = t_f\right] \\ \times pos\_mismatch\_penalty$$

We set $pos\_mismatch\_penalty$, a parameter that penalizes mismatching POS tags, to -2.[6]

This procedure can be used for almost all sentences with gapping constructions. However, if parts of an argument were elided along with the main predicate, it can become necessary to copy multiple nodes. We therefore consider 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 {*Sue*, *a book*}. In this case, we also consider the partial arguments {*Mary*, *a play*} and if the arguments of the gapped

[5]As suggested by one of the reviewers, we also ran a post-hoc experiment with a simpler similarity score function without the embedding distance term, which only takes into account whether the POS tags match. We found that quantitatively, the embeddings do not lead to significant better scores on the test set according to our metrics but qualitatively, they lead to better results for the examples with verb cluster gaps.

[6]We optimized this parameter on the training set by trying integer values from -1 to -15.

	EWT			GAPPING			EWT + GAPPING (COMBINED)		
	Train	**Dev**	**Test**	**Train**	**Dev**	**Test**	**Train**	**Dev**	**Test**
sentences	12,543	2,002	2,077	164	79	79	12,707	2,081	2,156
tokens	204,585	25,148	25,096	4,698	2,383	2,175	209,283	27,531	27,271
sentences with gapping	21 0.15%	1 0.05%	1 0.05%	164 100%	79 100%	79 100%	185 1.46%	80 3.84%	80 3.71%
copy nodes	22	2	1	201	96	102	223	98	103
unique composite relations	16	6	2	41	29	31	46	29	31

Table 1: Treebank statistics. The *copy nodes* row lists the number of copy nodes and the *unique composite 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.

Gap type	**Frequency**
Single predicate	172
Contiguous predicate-argument	140
Non-contiguous predicate-argument	9
Verb cluster	24

Table 2: Distribution of gap types in our corpus. The classification is according to the four types of gaps that we discussed in Section 2.1.

conjunct align better to the partial arguments, we use this alignment. However, now that the token *write* is part of the dependency path between *want* and *play*, we also have to make a copy of *write* to reconstruct the UD graph of the gapped clause.

4 Experiments

Both methods rely on a dependency parser followed by a post-processing step. We evaluated the individual steps and the end-to-end performance.

4.1 Data

We used the UD English Web Treebank v2.1 (henceforth EWT; Silveira et al., 2014; Nivre et al., 2017) for training and evaluating parsers. As the treebank is relatively small and therefore only contains very few sentences with gapping, we also extracted gapping constructions from the WSJ and Brown portions of the PTB (Marcus et al., 1993) and the GENIA corpus (Ohta et al., 2002). Further, we copied sentences from the Wikipedia page on gapping[7] and from published papers on gapping. The sentences in the EWT already contain annotations with the `orphan` relation and copy nodes for the enhanced representation, and we manually added both of these annotations for the remaining examples. The composite relations can be automatically obtained from the enhanced representation by removing the copy nodes and concatenating the dependency labels, which we did to build the training and test corpus for the composite relation procedure. Table 1 shows properties of the data splits of the original treebank, the additional sentences with gapping, and their combination; Table 2 shows the number of sentences in our corpus for each of the gap types.

4.2 Parsing experiments

Parser We used the parser by Dozat and Manning (2017) for parsing to the two different intermediate dependency representations. This parser is a graph-based parser (McDonald et al., 2005) that uses a biLSTM to compute token representations and then uses a multi-layer perceptron with biaffine attention to compute arc and label scores.

Setup We trained the parser on the COMBINED training corpus with gold tokenization, and predicted fine-grained and universal part-of-speech tags, for which we used the tagger by Dozat et al. (2017). We trained the tagger on the COMBINED training corpus. As pre-trained embeddings, we used the word2vec (Mikolov et al., 2013) embeddings that were provided for the CoNLL 2017 Shared Task (Zeman et al., 2017), and we used the same hyperparameters as Dozat et al. (2017).

Evaluation We evaluated the parseability of the two dependency representations using labeled and unlabeled attachment scores (LAS and UAS). Further, to specifically evaluate how well parsers are able to parse gapping constructions according to the two annotation schemes, we also computed the LAS and UAS just for the head tokens of remnants (LAS_g and UAS_g). For all our metrics, we excluded punctuation tokens. To determine sta-

[7] `https://en.wikipedia.org/wiki/Gapping`, accessed on Aug 24, 2017.

		EWT		GAPPING	
		UAS	LAS	UAS	LAS
Dev	ORPHAN	**90.57**	87.32	**89.34**	**85.69****
	COMPOSITE	90.46	**87.37**	88.86	84.21
Test	ORPHAN	90.42	87.06	**87.44**	**83.97****
	COMPOSITE	**90.54**	**87.33**	86.51	81.69

Table 3: Labeled (LAS) and unlabeled attachment score (UAS) of parsers trained and evaluated on the UD representation (ORPHAN) and the composite relations representation (COMPOSITE) on the development and test sets of the EWT and the GAPPING treebank. ** indicates that results differ significantly at $p < 0.01$.

	Development		Test	
	UAS$_g$	LAS$_g$	UAS$_g$	LAS$_g$
ORPHAN	**72.36**	**64.73*****	**72.56***	**65.79*****
COMPOSITE	68.36	49.45	62.41	46.24

Table 4: Labeled (LAS$_g$) and unlabeled attachment score (UAS$_g$) of head tokens of remnants for parsers trained and evaluated on the UD representation (ORPHAN) and the composite relations representation (COMPOSITE) on the development and test sets of the COMBINED treebank. Results that differ significantly are marked with * ($p < 0.05$) or *** ($p < 0.001$).

tistical significance of pairwise comparisons, we performed two-tailed approximate randomization tests (Noreen, 1989; Yeh, 2000) with an adapted version of the `sigf` package (Padó, 2006).

Results Table 3 shows the overall parsing results on the development and test sets of the two treebanks. There was no significant difference between the parser that was trained on the UD representation (ORPHAN) and the parser trained on the composite representation (COMPOSITE) when tested on the EWT data sets, which is not surprising considering that there is just one sentence with gapping each in the development and the test split. When evaluated on the GAPPING datasets, the ORPHAN parser performs significantly better ($p < 0.01$) in terms of labeled attachment score, which suggests that the parser trained on the COMPOSITE representation is indeed struggling with the greatly increased label space. This is further confirmed by the attachment scores of the head tokens of remnants (Table 4). The labeled attachment score of remnants is significantly higher for the ORPHAN parser than for the COMPOSITE parser. Further, the unlabeled attachment score on the test set is also higher for the ORPHAN parser, which suggests that the COMPOSITE parser is sometimes struggling with finding the right attachment for the multiple long-distance composite dependencies.

4.3 Recovery experiments

Our second set of experiments concerns the recovery of the elided material and the reattachment of the orphans. We conducted two experiments: an oracle experiment that used gold standard dependency trees and an end-to-end experiment that used the output of the parser as input. For all experiments, we used the COMBINED treebank.

Evaluation Here, we evaluated dependency graphs and therefore used the labeled and unlabeled precision and recall metrics. However, as our two procedures are only changing the attachment of orphans, we only computed these metrics for copy nodes and their dependents. Further, we excluded punctuation and coordinating conjunctions as their attachment is usually trivial and including them would inflate scores. Lastly, we computed the sentence-level accuracy for all sentences with gapping. For this metric, we considered a sentence to be correct if all copy nodes and their dependents of a sentence were attached to the correct head with the correct label.

Oracle results The top part of Table 5 shows the results for the oracle experiment. Both methods are able to reconstruct the elided material and the canonical clause structure from gold dependency trees with high accuracy. This was expected for the COMPOSITE procedure, which can make use of the composite relations in the dependency trees, but less so for the ORPHAN procedure which has to recover the structure and the types of relations. The two methods work equally well in terms of all metrics except for the sentence-level accuracy, which is significantly higher for the COMPOSITE procedure. This difference is caused by a difference in the types of mistakes. All errors of the COMPOSITE procedure are of a structural nature and stem from copying the wrong number of nodes while the dependency labels are always correct because they are part of the dependency tree. The majority of errors of the ORPHAN procedure stem from incorrect dependency labels, and these mistakes are scattered across more examples, which leads to the lower sentence-level accuracy.

End-to-end results The middle part of Table 5 shows the results for the end-to-end experiment. The performance of both methods is considerably lower than in the oracle experiment, which is pri-

		Development					Test				
		UP	UR	LP	LR	SAcc.	UP	UR	LP	LR	SAcc.
oracle	COMPOSITE	91.32	88.20	**91.32**	**88.20**	**91.14****	90.71	86.81	**90.71**	86.81	**86.08***
	ORPHAN	**94.08**	**93.79**	87.54	87.27	72.15	**92.02**	**92.02**	87.12	**87.12**	72.15
end-to-end	COMPOSITE	70.48	**49.69***	**65.64**	**46.27***	**31.65**	67.39	**47.55**	61.74	**43.56**	31.65
	ORPHAN	**71.73**	42.55	65.45	38.82	30.38	**78.92****	44.78	**68.11**	38.65	**34.18**
	K&K 2017	-	-	-	-	0.00	-	-	-	-	0.00

Table 5: Labeled and unlabeled precision and recall as well as sentence-level accuracy of the two gapping reconstructions methods and the K&K parser on the development and test set of the COMBINED treebank. Results that differ significantly from the other result within the same section are marked with * ($p < 0.05$) or ** ($p < 0.01$).

marily driven by the much lower recall. Both methods assume that the parser detects the existence of a gap and if the parser fails to do so, neither method attempts to reconstruct the elided material. In general, precision tends to be a bit higher for the ORPHAN procedure whereas recall tends to be a bit higher for the COMPOSITE method but overall and in terms of sentence-level accuracy both methods seem to perform equally well.

Error analysis For both methods, the primary issue is low recall, which is a result of parsing errors. When the parser correctly predicts the `orphan` relation, the main sources of error for the ORPHAN procedure are missing correspondents for remnants (e.g., *[for good]* has no correspondent in *They had left the company, many for good*) or that the types of argument of the remnant and its correspondent differ (e.g., in *She was convicted of selling unregistered securities in Florida and of unlawful phone calls in Ohio*, *[of selling unregistered securities]* is an adverbial clause whereas *[of unlawful phone calls]* is an oblique modifier).

Apart from the cases where the COMPOSITE procedure leads to an incorrect structure, the remaining errors are all caused by the parser predicting the wrong composite relation.

4.4 Comparison to Kummerfeld and Klein

Kummerfeld and Klein (henceforth K&K; 2017) recently proposed a one-endpoint-crossing graph parser that is able to directly parse to PTB-style trees with traces. They also briefly discuss gapping constructions and their parser tries to output the co-indexing that is used for gapping constructions in the PTB. The EWT and all the sentences that we took from the WSJ, Brown, and GENIA treebanks already come with constituency tree annotations, and we manually annotated the remaining sentences according to the PTB guidelines (Bies et al., 1995). This allowed us to train the K&K parser with exactly the same set of sentences that we used in our previous experiments. As this parser outputs constituency trees, we could not compute dependency graph metrics for this method. For the sentence-level accuracy, we considered an example to be correct if a) each argument in the gapped conjunct was the child of a single constituent node, which in return was the sibling of the full clause/verb phrase, and b) the co-indexing of each argument in the gapped conjunct was correct. For example, the following bracketing would be considered correct despite the incorrect internal structure of the first conjunct:

$[_{\text{S}}[_{\text{S}}[_{\text{NP-1}}$ Al] likes $[_{\text{NP-2}}$ coffee]] and $[_{\text{S}}[_{\text{NP=1}}$ Sue]$[_{\text{NP=2}}$ tea]]]

The last row of Table 5 shows the results of the K&K parser. The parser failed to output the correct constituency structure or co-indexing for every single example in the development and test sets. The parser struggled in particular with outputting the correct co-indices: For 32.5% of the test sentences with gapping, the bracketing of the gapped clause was correct but one or more of the co-indices were missing from the output.

Overall these results suggest that our dependency-based approach is much more reliable at identifying gapping constructions than the parser by K&K, which, in their defense, was optimized to output traces for other phenomena. Our method is also faster and took only seconds to parse the test set, while the K&K parser took several hours.

5 Resolving gaps in other languages

One of the appeals of the ORPHAN procedure is that it can be easily applied to other languages even if there exist no annotated enhanced dependency graphs.[8] On the one hand, this is because

[8]There is no theoretical reason that would prevent one from using the COMPOSITE procedure for other languages

our method does not make use of lexical information, and on the other hand, this is because we developed our method on top of the UD annotation scheme, which has already been applied to many languages and for which many treebanks exist.

Currently, all treebanks but the English one lack copy nodes for gapping constructions and many of them incorrectly use the `orphan` relation (Droganova and Zeman, 2017) and therefore we could not evaluate our method on a large variety of languages. In order to demonstrate that our method can be applied to other languages, we therefore did a case study on the Swedish UD treebank. The Swedish UD treebank is an automatic conversion from a section of the Talbanken (Einarsson, 1976) with extensive manual corrections. While the treebank is overall of high quality, we noticed conversion errors that led to incorrect uses of the `orphan` relation in 11 of the 29 sentences with `orphan` relations, which we excluded from our evaluation. We applied our gapping resolution procedure without any modifications to the remaining 18 sentences. We used the Swedish word2vec embeddings that were prepared for the CoNLL 2017 Shared Task. Our method correctly predicts the insertion of 29 copy nodes and is able to predict the correct structure of the enhanced representation in all cases, including complex ones with elided verb clusters such as the example in Figure 2. It also predicts the correct dependency label for 108/110 relations, leading to a labeled precision and labeled recall of 98.18%, which are both higher than the English numbers despite the fact that we optimized our procedure for English. The main reason for the higher performance seems to be that many of the Swedish examples come from informational texts from public organizations, which are more likely to be written to be clear and unambiguous. Further, the Swedish data does not contain challenging examples from the linguistic literature.

As Swedish is a Germanic language like English and thus shares many structural properties, we cannot conclude that our method is applicable to any language based on just this experiment. However, given that our method does not rely on language-specific structural patterns, we expect it to work well for a wide range of languages.

but given that UD treebanks are annotated with `orphan` relations, using the the COMPOSITE procedure would require additional manual annotations in practice.

6 Related work

Gapping constructions have been little studied in NLP, but several approaches (e.g., Dukes and Habash 2011; Simkó and Vincze 2017) parse to dependency trees with empty nodes. Seeker et al. (2012) compared three ways of parsing with empty heads: adding a transition that inserts empty nodes, using composite relation labels for nodes that depend on an elided node, and pre-inserting empties before parsing. These papers all focus on recovering nodes for elided function words such as auxiliaries; none of them attempt to recover and resolve the content word elisions of gapping. Ficler and Goldberg (2016) modified PTB annotations of argument-cluster coordinations (ACCs), i.e., gapping constructions with two post-verbal orphan phrases, which make up a subset of the gapping constructions in the PTB. While the modified annotation style leads to higher parsing accuracy of ACCs, it is specific to ACCs and does not generalize to other gapping constructions. Moreover, they did not reconstruct gapped ACC clauses. Traditional grammar-based chart parsers (Kay, 1980; Klein and Manning, 2001) did handle empty nodes and so could in principle provide a parse of gapping sentences though additional mechanisms would be needed for reconstruction. In practice, though, dealing with gapping in a grammar-based framework is not straightforward and can lead to a combinatorial explosion that slows down parsing in general, as has been noted for the English Resource Grammar (Flickinger, 2017, p.c.) and for an HPSG implementation for Norwegian (Haugereid, 2017). The grammar-based parser built with augmented transition networks (Woods, 1970) provided an extension in the form of the SYSCONJ operation (Woods, 1973) to parse some gapping constructions, but also this approach lacked explicit reconstruction mechanisms and provided only limited coverage.

There also exists a long line of work on post-processing surface-syntax constituency trees to recover traces in the PTB (Johnson, 2002; Levy and Manning, 2004; Campbell, 2004; Gabbard et al., 2006), pre-processing sentences such that they contain tokens for traces before parsing (Dienes and Dubey, 2003b), or directly parsing sentences to either PTB-style trees with empty elements or pre-processed trees that can be deterministically converted to PTB-style trees (Collins,

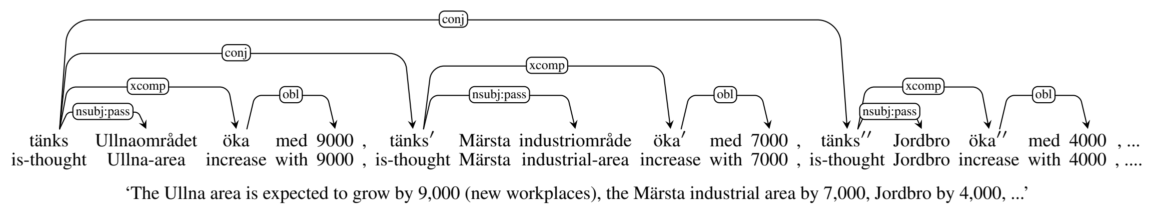

conj
conj
xcomp
xcomp
xcomp
nsubj:pass
nsubj:pass
nsubj:pass
obl
obl
obl
tänks Ullnaområdet öka med 9000 , tänks′ Märsta industriområde öka′ med 7000 , tänks′′ Jordbro öka′′ med 4000 , ...
is-thought Ullna-area increase with 9000 , is-thought Märsta industrial-area increase with 7000 , is-thought Jordbro increase with 4000 , ...
'The Ullna area is expected to grow by 9,000 (new workplaces), the Märsta industrial area by 7,000, Jordbro by 4,000, ...'

Figure 2: Dependency graph for part of the sentence `sv-ud-train-1102` as output by the ORPHAN procedure. The system correctly predicts the copy nodes for the matrix and the embedded verb, and correctly attaches the arguments to the copy nodes.

1997; Dienes and Dubey, 2003a; Schmid, 2006; Cai et al., 2011; Hayashi and Nagata, 2016; Kato and Matsubara, 2016; Kummerfeld and Klein, 2017). However, all of these works are primarily concerned with recovering traces for phenomena such as Wh-movement or control and raising constructions and, with the exception of Kummerfeld and Klein (2017), none of these works attempt to output the co-indexing that is used for analyzing gapping constructions. And again, none of these works try to reconstruct elided material.

Lastly, several methods have been proposed for resolving other forms of ellipsis, including VP ellipsis (Hardt, 1997; Nielsen, 2004; Lappin, 2005; McShane and Babkin, 2016) and sluicing (Anand and Hardt, 2016) but none of these methods consider gapping constructions.

7 Conclusion

We presented two methods to recover elided predicates in sentences with gapping. Our experiments suggest that both methods work equally well in a realistic end-to-end setting. While in general, recall is still low, the oracle experiments suggest that both methods can recover elided predicates from correct dependency trees, which suggests that as parsers become more and more accurate, the gap recovery accuracy should also increase.

We also demonstrated that our method can be used to automatically add the enhanced UD representation to UD treebanks in other languages than English. Apart from being useful in a parsing pipeline, we therefore also expect our method to be useful for building enhanced UD treebanks.

Reproducibility

All data, pre-trained models, system outputs as well as a package for running the enhancement procedure are available from `https://github.com/sebschu/naacl-gapping`.

Acknowledgments

We thank the anonymous reviewers for their thoughtful feedback. Also thanks to Vera Gribanova and Boris Harizanov for continuous feedback throughout this project, and to Matthew Lamm for help with annotating the data. This work was supported in part by gifts from Google, Inc. and IPSoft, Inc. The first author is also supported by a Goodan Family Graduate Fellowship.

References

Pranav Anand and Daniel Hardt. 2016. Antecedent selection for sluicing: Structure and content. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*. pages 1234–1243. `https://doi.org/10.18653/v1/D16-1131`.

Eduard Bejček, Eva Hajičová, Jan Hajič, Pavlína Jínová, Václava Kettnerová, Veronika Kolářová, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Jarmila Panevová, Lucie Poláková, Magda Ševčíková, Jan Štěpánek, and 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`.

Ann Bies, Mark Ferguson, Karen Katz, Robert MacIntyre, Victoria Tredinnick, Grace Kim, Mary Ann Marcinkiewicz, and Britta Schasberger. 1995. Bracketing guidelines for Treebank II style Penn Treebank project. Technical report, University of Pennsylvania.

Shu Cai, David Chiang, and Yoav Goldberg. 2011. Language-independent parsing with empty elements. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL 2011)*. pages 212–216. `http://www.aclweb.org/anthology/P11-2037`.

Richard Campbell. 2004. Using linguistic principles to recover empty categories. In *Proceedings of the*

42nd Annual Meeting on Association for Computational Linguistics (ACL 2004). https://doi.org/10.3115/1218955.1219037.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL 1997)*. pages 16–23. https://doi.org/10.3115/976909.979620.

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

Péter Dienes and Amit Dubey. 2003a. Antecedent recovery: Experiments with a trace tagger. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP 2003)*. pages 33–40. http://www.aclweb.org/anthology/W03-1005.

Péter Dienes and Amit Dubey. 2003b. Deep syntactic processing by combining shallow methods. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL 2003)*. pages 431–438. https://doi.org/10.3115/1075096.1075151.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*. pages 1–8. https://openreview.net/pdf?id=Hk95PK9le.

Timothy Dozat, Peng Qi, and 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*. pages 20–30. https://doi.org/10.18653/v1/K17-3002.

Kira Droganova and Daniel Zeman. 2017. Elliptic constructions: Spotting patterns in UD treebanks. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*. pages 48–57. http://www.aclweb.org/anthology/W17-0406.

Kais Dukes and Nizar Habash. 2011. One-step statistical parsing of hybrid dependency-constituency syntactic representations. In *Proceedings of the 12th International Conference on Parsing Technologies*. pages 92–103. http://www.aclweb.org/anthology/W11-2912.

Jan Einarsson. 1976. Talbankens skriftspråkskonkordans. Institutionen för nordiska språk, Lunds universitet.

Annahita Farudi. 2013. *Gapping in Farsi: A Crosslinguistic Investigation*. Ph.D. thesis, University of Massachusetts Amherst. http://scholarworks.umass.edu/dissertations/AAI3556244.

Jessica Ficler and Yoav Goldberg. 2016. Improved parsing for argument-clusters coordination. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*. pages 72–76. https://doi.org/10.18653/v1/P16-2012.

Ryan Gabbard, Mitchell Marcus, and Seth Kulick. 2006. Fully parsing the Penn Treebank. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL (NAACL 2006)*. pages 184–191. https://doi.org/10.3115/1220835.1220859.

Daniel Hardt. 1997. An empirical approach to VP ellipsis. *Computational Linguistics* 23(4):525–541. http://www.aclweb.org/anthology/J97-4002.

Petter Haugereid. 2017. An incremental approach to gapping and conjunction reduction. In *Proceedings of the 24th International Conference on Head-Driven Phrase Structure Grammar*. pages 179–198. http://cslipublications.stanford.edu/HPSG/2017/hpsg2017-haugereid.pdf.

Katsuhiko Hayashi and Masaaki Nagata. 2016. Empty element recovery by spinal parser operations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*. pages 95–100. https://doi.org/10.18653/v1/P16-2016.

Julia Hockenmaier and Mark Steedman. 2007. CCG-bank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics* 33(3):355–396. http://www.aclweb.org/anthology/J07-3004.

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

Kyle Johnson. 2014. Gapping. Unpublished manuscript, University of Massachusetts at Amherst. http://people.umass.edu/kbj/homepage/Content/gapping.pdf.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL 2002)*. pages 136–143. https://doi.org/10.3115/1073083.1073107.

Yoshihide Kato and Shigeki Matsubara. 2016. Transition-based left-corner parsing for identifying PTB-style nonlocal dependencies. In *Proceedings of the 54th Annual Meeting of* *the Association for Computational Linguistics (ACL 2016)*. pages 930–940. https://doi.org/10.18653/v1/P16-1088.

Martin Kay. 1980. Algorithm schemata and data structures in syntactic processing. Technical report, Xerox PARC.

Dan Klein and Christopher Manning. 2001. Parsing with treebank grammars: empirical bounds, theoretical models, and the structure of the Penn Treebank. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics (ACL 2001)*. pages 338–345. https://doi.org/10.3115/1073012.1073056.

Yusuke Kubota and Robert Levine. 2016. Gapping as hypothetical reasoning. *Natural Language and Linguistic Theory* 34(1):107–156. https://doi.org/10.1007/s11049-015-9298-4.

Jonathan K. Kummerfeld and Dan Klein. 2017. Parsing with traces: An $O(n^4)$ algorithm and a structural representation. *Transactions of the Association for Computational Linguistics* 5:441–454. https://www.transacl.org/ojs/index.php/tacl/article/view/1170.

Shalom Lappin. 2005. A sequenced model of anaphora and ellipsis resolution. In António Branco, Tony McEnery, and Ruslan Mitkov, editors, *Anaphora Processing: Linguistic, Cognitive and Computational Modelling*, John Benjamins Publishing, Amsterdam, pages 3–16. https://doi.org/10.1075/cilt.263.03lap.

Roger Levy and Christopher D. Manning. 2004. Deep dependencies from context-free statistical parsers. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL 2004)*. pages 327–334. https://doi.org/10.3115/1218955.1218997.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* 19(2):313–330. http://www.aclweb.org/anthology/J93-2004.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT-EMNLP 2005)*. pages 523–530. https://doi.org/10.3115/1220575.1220641.

Marjorie McShane and Petr Babkin. 2016. Detection and resolution of verb phrase ellipsis. *Linguistic Issues in Language Technology (LiLT)* 13(1):1–34.

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

Saul B. Needleman and 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. https://doi.org/10.1016/0022-2836(70)90057-4.

Leif Arda Nielsen. 2004. Verb phrase ellipsis detection using automatically parsed text. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*. 1, pages 1093–1099. https://doi.org/10.3115/1220355.1220512.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Lene Antonsen, Maria Jesus Aranzabe, Masayuki Asahara, Luma Ateyah, Mohammed Attia, Aitziber Atutxa, Liesbeth Augustinus, Elena Badmaeva, Miguel Ballesteros, Esha Banerjee, Sebastian Bank, Verginica Barbu Mititelu, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Victoria Bobicev, Carl Börstell, Cristina Bosco, Gosse Bouma, Sam Bowman, Aljoscha Burchardt, Marie Candito, Gauthier Caron, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Savas Cetin, Fabricio Chalub, Jinho Choi, Silvie Cinková, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Peter Dirix, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Ali Elkahky, Tomaž Erjavec, Richárd Farkas, Hector Fernandez Alcalde, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Moa Gärdenfors, Kim Gerdes, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Jan Hajič jr., Linh Hà Mỹ, Kim Harris, Dag Haug, Barbora Hladká, Jaroslava Hlaváčová, Florinel Hociung, Petter Hohle, Radu Ion, Elena Irimia, Tomáš Jelínek, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Tolga Kayadelen, Václava Kettnerová, Jesse Kirchner, Natalia Kotsyba, Simon Krek, Veronika Laippala, Lorenzo Lambertino, Tatiana Lando, John Lee, Phương Lê Hồng, Alessandro Lenci, Saran Lertpradit, Herman Leung, Cheuk Ying Li, Josie Li, Keying Li, Nikola Ljubešić, Olga Loginova, Olga Lyashevskaya, Teresa Lynn, Vivien Macketanz, Aibek Makazhanov, Michael Mandl, Christopher D. Manning, Cătălina Mărănduc, David Mareček, Katrin Marheinecke, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Gustavo Mendonça, Niko Miekka, Anna Missilä, Cătălin Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Laura Moreno Romero, Shinsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Kaili Müürisep, Pinkey Nainwani, Anna Nedoluzhko, Gunta Nešpore-Bērzkalne, Lương Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Robert Östling, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Emily Pitler, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Taraka Rama, Vinit Ravishankar, Livy Real, Siva Reddy, Georg Rehm, Larissa Rinaldi, Laura Rituma, Mykhailo Romanenko, Rudolf Rosa, Davide Rovati, Benoît Sagot, Shadi Saleh, Tanja Samardžić, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Mo Shen, Atsuko Shimada, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Antonio Stella, Milan Straka, Jana Strnadová, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Trond Trosterud, Anna Trukhina, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Zdeňka Urešová, Larraitz Uria, Hans Uszkoreit, Sowmya Vajjala, Daniel van Niekerk, Gertjan van Noord, Viktor Varga, Eric Villemonte de la Clergerie, Veronika Vincze, Lars Wallin, Jonathan North Washington, Mats Wirén, Tak-sum Wong, Zhuoran Yu, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal Dependencies 2.1. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University. `http://hdl.handle.net/11234/1-2515`.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. pages 1659–1666. `http://www.lrec-conf.org/proceedings/lrec2016/pdf/348_Paper.pdf`.

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

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinkova, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. SemEval 2015 Task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. pages 915–926. `https://doi.org/10.18653/v1/S15-2153`.

Tomoko Ohta, Yuka Tateisi, and 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 (HLT 2002)* pages 82–86. `https://doi.org/10.3115/1289189.1289260`.

Timothy Osborne. 2006. Gapping vs. non-gapping coordination. *Linguistische Berichte* 207:307–337.

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

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*. pages 1532–1543. `https://doi.org/10.3115/v1/D14-1162`.

John Robert Ross. 1970. Gapping and the order of constituents. In Manfred Bierwisch and Karl Erich Heidolph, editors, *Progress in Linguistics*, De Gruyter, The Hague, pages 249–259.

Helmut Schmid. 2006. Trace prediction and recovery with unlexicalized PCFGs and slash features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL (ACL 2006)*. pages 177–184. `https://doi.org/10.3115/1220175.1220198`.

Sebastian Schuster, Matthew Lamm, and Christopher D. Manning. 2017. Gapping constructions in Universal Dependencies v2. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*. pages 123–132. `http://www.aclweb.org/anthology/W17-0416`.

Sebastian Schuster and 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)*. pages 2371–2378. `http://www.lrec-conf.org/proceedings/lrec2016/pdf/779_Paper.pdf`.

Wolfgang Seeker, Richárd Farkas, Bernd Bohnet, Helmut Schmid, and Jonas Kuhn. 2012. Data-driven dependency parsing with empty heads. In *Proceedings of COLING 2012*. pages 1081–1090. `http://www.aclweb.org/anthology/C12-2105`.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and 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)*. pages 2897–2904. `http://www.lrec-conf.org/proceedings/lrec2014/pdf/1089_Paper.pdf`.

Katalin Ilona Simkó and Veronika Vincze. 2017. Hungarian copula constructions in dependency syntax and parsing. In *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017)*. pages 240–247. `http://www.aclweb.org/anthology/W17-6527`.

Mark Steedman. 1990. Gapping as constituent coordination. *Linguistics and Philosophy* 13(2):207–263. https://doi.org/10.1007/BF00630734.

Maziar Toosarvandani. 2016. Embedding the antecedent in gapping: Low coordination and the role of parallelism. *Linguistic Inquiry* 47(2):381–390. https://doi.org/10.1162/LING_a_00216.

William A. Woods. 1970. Transition network grammars for natural language analysis. *Communications of the ACM* 13(10):591–606.

William A. Woods. 1973. An experimental parsing system for transition network grammars. In Randall Rustin, editor, *Natural Language Processing*, Algorithmics Press, New York, pages 111–154.

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

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the The 18th International Conference on Computational Linguistics (COLING 2000)*. pages 947–953. http://www.aclweb.org/anthology/C00-2137.

Daniel Zeman, Martin Popel, Milan Straka, Jan Hajič, Joakim Nivre, Filip Ginter, Juhani Luotolahti, Sampo Pyysalo, Slav Petrov, Martin Potthast, Francis Tyers, Elena Badmaeva, Memduh Gökrmak, 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, and 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*. pages 1–19. https://doi.org/10.18653/v1/K17-3001.