Parsing Gapping Constructions Based on Grammatical and Semantic Roles

Yoshihide Kato and **Shigeki Matsubara**
Information & Communications, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, 464-8601 Japan
yoshihide@icts.nagoya-u.ac.jp

Abstract

A gapping construction consists of a coordinated structure where redundant elements are elided from all but one conjuncts. This paper proposes a method of parsing sentences with gapping to recover elided elements. The proposed method is based on constituent trees annotated with grammatical and semantic roles that are useful for identifying elided elements. Our method outperforms the previous method in terms of F-measure and recall.

1 Introduction

A gapping construction consists of a coordinated structure where redundant elements are elided from all but one conjuncts. For example, we can elide the second redundant verb "ate" from the sentence "John ate bread, and Mary ate rice." We need to recover elided elements to interpret sentences with gapping; however, little work has focused on developing such methods.

This paper proposes a method of parsing sentences with gapping. Our proposed method uses constituent trees annotated with grammatical and semantic tags and a special tag indicating gapped conjuncts. The method parses a sentence to obtain a tag-annotated constituent tree and analyzes gapping constructions using the resulting tree when it includes gapped conjuncts. The analysis is based on a sequence alignment algorithm using grammatical and semantic tags. An experiment shows that our method outperforms the previous method in terms of F-measure and recall.

2 Gapping Construction

This section first explains gapping constructions in the Penn Treebank (PTB, Marcus et al., 1993), on which our proposed method is based, and summarizes the previous work on analyzing sentences with gapping.

2.1 Gapping Constructions in the PTB

A gapping construction consists of a coordinated structure where redundant elements are elided from all but one conjuncts. The constituents remaining in a gapped conjunct are called *remnants*. The remnants have a corresponding constituent, called a *correlate*, in the ungapped conjunct.[1] We can obtain the ungapped version of the conjunct by replacing each correlate with its corresponding remnant.

In the PTB, the correspondences between remnants and correlates are annotated. Figure 1 shows an example of a PTB constituent tree, which includes a gapping construction. The nodes marked with "-" hyphen indices are the correlates, while those marked with "=" equal indices are the remnants. A gapped conjunct is flattened, that is, all remnants are children of the conjunct node. The number assigned to a correlate and a remnant indicates a correspondence relation. For example, `NP-SBJ-1` and `NP-SBJ=1` in this tree are a correlate and a remnant, respectively, and correspond to each other. We can obtain the constituent tree for "the six-month bills will still mature on May 3, 1990" by replacing `NP-SBJ-1` with the tree whose root is `NP-SBJ=1` and `NP-TMP-2` with that whose root is `PP-TMP=2`. In other words, "will still mature" is elided from the second conjunct.

2.2 Previous Work

This section gives an overview of previous approaches to analyzing sentences with gapping.

Ficler and Goldberg (2016) proposed a new representation for argument-cluster coordination, which is one kind of gapping constructions. They converted PTB trees by coordinating correlates and remnants. This conversion can be applied only when the correlates and the remnants are all to-

[1] In English, the first conjunct is ungapped.

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 2747–2752,
November 16–20, 2020.

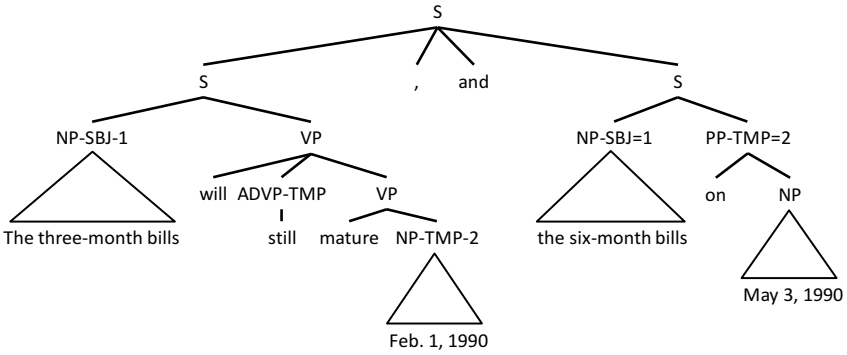

Figure 1: Gapping construction in the PTB.

gether on the right. Therefore, it cannot handle the tree shown in Figure 1.

Kummerfeld and Klein (2017) developed a parser that adopts a graph representation for syntactic structure. They discussed how to represent a correspondence between a remnant and a correlate with an arc in their graph representation. However, the parser struggled to generate such arcs, and the recall was very low.[2]

Schuster et al. (2018) proposed two methods based on dependency structure. One represents gapping constructions using complex relation labels (Seeker et al., 2012), and the other adopts a sequence alignment algorithm to assign remnant words to correlate words. The latter is similar to ours. We will discuss the differences between the latter method and ours in the later section.

Another approach is the one that does not depend on syntactic representation. In the Automatic Gapping Resolution Shared Task for Russian (AGRR-2019) (Ponomareva et al., 2019), the prepared dataset marked each element comprising gapping constructions. Most participants treated this task as a sequence labeling problem.

3 Proposed Method

This section describes our proposed method, which parses gapping constructions. Our method enables existing PTB-based parsers to identify correspondences between correlates and remnants. Elided elements can be recovered by such correspondences as described in Section 2.1.

One difficulty in parsing gapping constructions is insufficient data for modeling such phenomena because their occurrence is rare. To mitigate this problem, our method learns not directly from correspondences between correlates and remnants, but from the following tags easily obtained from the PTB:

- a special tag indicating gapped conjuncts
- grammatical and semantic role tags

Correspondences between correlates and remnants are identified by a sequence alignment algorithm using the tag-annotated constituent tree. We first explain our tag annotation and describe the sequence alignment algorithm.

3.1 Annotation

Our method uses a special tag to identify gapped conjuncts in constituency parsing. Specifically, we assign the `GAP` tag to a node n, if n satisfies the following condition:

- n has a child marked with "=" index.

In the coordinated structure shown in Figure 1, the `GAP` tag is assigned to the second `S` conjunct node.

Next, we explain grammatical and semantic tags. In general, each remnant and its corresponding correlate play an identical grammatical or semantic role. For example, in Figure 1, the constituents co-indexed with `1` are subjects (`SBJ`), while those co-indexed with `2` are temporal adjuncts (`TMP`). This fact suggests that correspondences between correlates and remnants can be identified using grammatical and semantic roles. Our method directly uses the PTB grammatical and semantic role tags shown in Table 1.[3]

An important point herein is that our method has an advantage from the viewpoint of training data. The PTB includes a massive amount of

[2]Note that the main purpose of the parser is to analyze traces, such as wh-movement, and not gapping constructions.

[3]The grammatical tag `PRD` can be used together with a semantic tag. We only used the `PRD` tag in this case.

grammatical		semantic	
DTV	Dative	BNF	Benefactive
LGS	Logical subject	DIR	Direction
PRD	Predicate	EXT	Extent
PUT	Locative complement of 'put'	LOC	Locative
SBJ	Surface subject	MNR	Manner
		PRP	Purpose
		TMP	Temporal

Table 1: Grammatical and semantic role tags.

grammatical and semantic tag information; thus, we can learn a model that identifies the tags by simply retaining them in the treebank (Gabbard et al., 2006).

3.2 How to identify correspondences between correlates and remnants

This section explains how to identify correspondences between correlates and remnants using the tag-annotated constituent trees described in the previous section. The procedure consists of the following two steps:

1. Extract remnant candidates R and correlate candidates C when gapped conjuncts exist.

2. Align nodes in R to nodes in C.

The first step is invoked if a node $n_{\mathtt{g}}$ annotated with the GAP tag exists. A set R of remnant candidates is defined as a set of $n_{\mathtt{g}}$'s children. To extract the correlate candidates C, the method seeks the ungapped conjunct $n_{\mathtt{u}}$ that satisfies the following condition:

$$\begin{aligned} & n_{\mathtt{u}} \in \mathtt{L}(n_{\mathtt{g}}) \\ & \wedge\, \mathtt{ct}(n_{\mathtt{u}}) = \mathtt{ct}(n_{\mathtt{g}}) \\ & \wedge\, \neg\exists n \in \mathtt{L}(n_{\mathtt{u}}) \big(\mathtt{ct}(n) = \mathtt{ct}(n_{\mathtt{u}})\big) \end{aligned}$$

where, $\mathtt{L}(n)$ and $\mathtt{ct}(n)$ are the set of left siblings of n and the category of n, respectively. A set C of correlate candidates is defined as a set of $n_{\mathtt{u}}$'s proper descendants.

The second step aligns nodes in R to nodes in C. Here we impose a constraint that $r \in R$ plays an identical role to $c \in C$. More precisely, we can align r to c if the following condition holds:

$$\begin{aligned} & \mathtt{match}(c, r) \overset{def}{=} \\ & \quad \big(\mathtt{rl}(c) = \mathtt{rl}(r) \neq \mathtt{null}\big) \\ & \quad \vee \big(\mathtt{ct}(c) = \mathtt{ct}(r) = \mathtt{PP} \wedge \mathtt{hd}(c) = \mathtt{hd}(r)\big) \\ & \quad \vee \big(\mathtt{rl}(c) = \mathtt{rl}(r) = \mathtt{null} \wedge \mathtt{ct}(c) = \mathtt{ct}(r)\big) \end{aligned}$$

where, $\mathtt{rl}(x)$ stands for the role tag of x. If x has no role tag, $\mathtt{rl}(x) = \mathtt{null}$. $\mathtt{hd}(x)$ is the head preposition of x. Furthermore, we impose the following structural constraints to follow the PTB annotation scheme:

Uniqueness of remnant If $(c, r) \in A$ and $(c, r') \in A$, then $r = r'$.

Uniqueness of correlate If $(c, r) \in A$ and $(c', r) \in A$, then $c = c'$.

Order-Preserving For all $(c, r), (c', r') \in A$, if $\mathtt{e}(r) \leq \mathtt{s}(r')$, then $\mathtt{e}(c) \leq \mathtt{s}(c')$.

Non-overlapping For all $(c, r), (c', r') \in A (c \neq c')$, then $\mathtt{e}(c) \leq \mathtt{s}(c')$ or $\mathtt{e}(c') \leq \mathtt{s}(c)$.

Here, $A \subseteq C \times R$ is a set representing an alignment of correlates to remnants. $(c, r) \in A$ means that a correlate c is aligned to a remnant r. $\mathtt{s}(n)$ and $\mathtt{e}(n)$ stand for the start and end positions of node n, respectively.

3.3 DP-based sequence alignment

To realize the second step described in Section 3.2, we modify the sequence alignment algorithm proposed by Needleman and Wunsch (1970). Our algorithm is shown in Algorithm 1. $\mathtt{T}[i, j]$ keeps the highest scoring alignment of $c_i \cdots c_m$ to $r_j \cdots r_n$ and its score. The difference between our modified version and the original one can be seen in line 10. While the original version considers the next element c_{i+1}, ours skips all the descendants of c_i. That is, $\mathtt{skip\text{-}des}[i]$ is the index such that $c_{\mathtt{skip\text{-}des}[i]}$ is the first non-descendant of c_i in $c_{i+1} \cdots c_m$. This modification is required to satisfy the order-preserving and non-overlapping constraints.[4] The function $\mathtt{score}$ is defined as follows:[5]

$$\mathtt{score}(c, r) = \begin{cases} 1 & (\mathtt{match}(c, r) = \mathtt{true}) \\ -\infty & (\mathtt{match}(c, r) = \mathtt{false}) \end{cases}$$

[4] C is sorted in a preorder traversal order, hence, all $c_k (i < k < \mathtt{skip\text{-}des}[i])$ are c_i's descendants, and all $c_l (\mathtt{skip\text{-}des}[i] \leq l \leq m)$ satisfy the equation $\mathtt{e}(c_i) \leq \mathtt{s}(c_l)$. That is, any c_k violates the constraints, and any c_l satisfies them.

[5] When two alignments obtain the same score, we prefer the one whose correlates cover more words.

Algorithm 1 Sequence alignment algorithm

Input: lists of correlate candidates $C = c_1 \cdots c_m$ and remnant candidates $R = r_1 \cdots r_n$. C and R are sorted in preorder traversal order.
for $i = 0$ **to** m **do**
  $\mathtt{T}[i, n] \leftarrow \langle \{\}, 0 \rangle$
end for
for $j = 0$ **to** n **do**
  $\mathtt{T}[m, j] \leftarrow \langle \{\}, 0 \rangle$
end for
for $i = m - 1$ **down to** 0 **do**
  for $j = n - 1$ **down to** 0 **do**
    $\langle A, s \rangle \leftarrow \mathtt{T}[\mathtt{skip\text{-}des}[i], j + 1]$
    $\mathtt{Match} \leftarrow \langle A \cup \{(c_i, r_j)\}, s + \mathtt{score}(c_i, r_j) \rangle$
    $\mathtt{T}[i, j] \leftarrow \underset{T \in \{\mathtt{T}[i+1,j], \mathtt{T}[i,j+1], \mathtt{Match}\}}{\arg\max} \mathtt{Score}(T)$
  end for
end for
return $T[0, 0]$

3.4 Discussion

Our method is similar to that of Schuster et al. (2018) in that both rely on sequence alignment. The following differences, however, exist:

- The previous method converts Universal Dependencies (UD) v2 representations (Nivre et al., 2017) to enhanced UD representations (Schuster and Manning, 2016), which provides an analysis of gapping constructions. It cannot use grammatical and semantic roles unlike ours, because remnants cannot have such information in the UD v2 framework.[6]
- Our proposed sequence alignment algorithm is a novel one that can impose the order-preserving and non-overlapping constraints on the resulting alignment. This feature is required to deal with gapping constructions in PTB constituent trees.

4 Experiment

We conducted an experiment using the PTB to evaluate the performance of our proposed method.[7] We used the standard PTB training, development, and test data split (i.e., sections 02–21, 22, and 23, respectively) and the Kitaev and Klein parser (Kitaev and Klein, 2018)[8] that can use BERT (Devlin et al., 2019). We trained the parsing model by simply replacing the training and development data with those annotated by our annotation scheme. The hyperparameters were identical to those of Kitaev et al. (2019). The test data were parsed by the trained model to obtain tag-annotated trees. The correspondences between the correlates and the remnants were identified by our proposed alignment algorithm. The alignment accuracy was evaluated by the metric of Kummerfeld and Klein (2017). That is, we represent a correspondence between a correlate c and a remnant r as a tuple $(\mathtt{ct}(r), \mathtt{s}(r), \mathtt{e}(r), \mathtt{ct}(c), \mathtt{s}(c), e(c))$, and measure the precision and recall using tuples.

	pre.	rec.	F
Kummerfeld and Klein (2017)	100.0	6.9	12.9
Ours (w/o BERT)	66.7	20.7	31.6
Ours (with BERT)	89.5	58.6	70.8
Ours (oracle)	92.6	86.2	89.3

Table 2: Alignment performance on the test data.

		w/o BERT			with BERT		
Tag	freq.	pre.	rec.	F1	pre.	rec.	F1
GAP	16	66.7	25.0	36.4	84.6	68.8	75.9
SBJ	4148	97.4	96.8	97.1	98.1	98.1	98.1
PRD	1025	82.0	78.8	80.4	88.3	85.1	86.6
LGS	166	88.6	88.6	88.6	91.5	90.4	90.9
DTV	19	73.7	73.7	73.7	87.5	73.7	80.0
PUT	10	50.0	40.0	44.4	72.7	80.0	76.2
TMP	1302	89.7	91.9	90.7	91.8	94.3	93.0
LOC	953	88.4	80.9	84.5	91.2	84.5	87.7
DIR	293	71.1	45.4	55.4	82.8	62.5	71.2
PRP	204	76.9	55.4	64.4	84.3	71.1	77.1
MNR	178	76.2	77.5	76.9	72.5	79.8	75.9
EXT	105	87.5	80.0	83.6	88.9	83.8	86.3
BNF	2	0.0	0.0	0.0	0.0	0.0	0.0

Table 3: Accuracy of tag identification on the test data.

Table 2 shows the alignment performance of our method and the previous one. The previous method struggled to generate arcs between correlates and remnants and had a very low recall. In contrast, our method achieved high recall and F-measure. It outperformed the previous method even without BERT.

The last row in Table 2 shows the performance when using gold tag-annotated trees, indicating that our sequence alignment algorithm works well and that tag identification directly affects the overall performance. We evaluated the tag identification accuracy using a tuple $(\mathtt{rl}(n), \mathtt{s}(n), \mathtt{e}(n))$. Table 3 presents the tag identification accuracy. The performance of identifying grammatical and semantic tags did not differ much between using BERT and not using BERT. On the other hand, the performance of identifying the GAP tag without BERT is rather low compared to that with

[6] In the UD v2 framework, one remnant is treated as a head, and the others are attached to it with the special orphan dependency.

[7] The code is available at https://github.com/yosihide/ptb2cf.

[8] https://github.com/nikitakit/self-attentive-parser

	pre.	rec.	F1
w/o tag annotation (w/o BERT)	93.90	93.20	93.55
w/o tag annotation (with BERT)	96.03	95.51	95.77
with tag annotation (w/o BERT)	93.65	92.79	93.22
with tag annotation (with BERT)	95.93	95.35	95.64

Table 4: Comparison for constituency parsing performance on the test data.

BERT. Therefore, the main reason for the low performance of the alignment without BERT is the degraded performance of identifying the GAP tag.

Finally, we report the constituency parsing performance. Table 4 shows the accuracy of the Kitaev and Klein parser with and without our tag annotation. The result implies that our tag annotation to constituent trees has a tiny negative impact on the constituency parsing performance.

5 Conclusion

This paper has proposed a method of parsing gapping constructions based on tag-annotated constituent trees. Our proposed method is simple but effective. We believe that it will serve as a strong baseline for the task of parsing gapping constructions. In the future work, we will extend our method by replacing the simple role matching score with grammatical or semantic similarity-based measures to improve the alignment accuracy.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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 (Volume 2: Short Papers)*, pages 72–76, Berlin, Germany. Association for Computational Linguistics.

Ryan Gabbard, Seth Kulick, and Mitchell Marcus. 2006. Fully parsing the Penn Treebank. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 184–191, New York City, USA. Association for Computational Linguistics.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

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.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):310–330.

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.

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 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/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Maria Ponomareva, Kira Droganova, Ivan Smurov, and Tatiana Shavrina. 2019. AGRR 2019: Corpus for gapping resolution in Russian. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 35–43, Florence, Italy. Association for Computational Linguistics.

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'16)*, pages 2371–2378, Portorož, Slovenia. European Language Resources Association (ELRA).

Sebastian Schuster, Joakim Nivre, and Christopher D. Manning. 2018. Sentences with gapping: Parsing and reconstructing elided predicates. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1156–1168, New Orleans, Louisiana. Association for Computational Linguistics.

Wolfgang Seeker, Richárd Farkas, Bernd Bohnet, Helmut Schmid, and Jonas Kuhn. 2012. Data-driven dependency parsing with empty heads. In *Proceedings of COLING 2012: Posters*, pages 1081–1090, Mumbai, India. The COLING 2012 Organizing Committee.