Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 5282–5290
Marseille, 11–16 May 2020
© European Language Resources Association (ELRA), licensed under CC-BY-NC

Recognizing Sentence-level Logical Document Structures with the Help of Context-free Grammars

Jonathan Hildebrand[1], Wahed Hemati2, Alexander Mehler[3]
[1,2,3]Text Technology Lab
[1,2,3]Goethe-Universität Frankfurt
[1]mail@jonathanhildebrand.de, [2]hemati@em.uni-frankfurt.de, [3]mehler@em.uni-frankfurt.de

Abstract

Current sentence boundary detectors, such as (Azzi et al., 2019; González-Gallardo and Torres-Moreno, 2018), split documents into sequentially ordered sentences by detecting their beginnings and ends. Sentences, however, are more deeply structured even on this side of constituent and dependency structure: they can consist of a main sentence and several subordinate clauses as well as further segments (e.g. inserts in parentheses); they can even recursively embed whole sentences and then contain multiple sentence beginnings and ends. In this paper, we introduce a tool that segments sentences into tree structures to detect this type of recursive structure. To this end, we retrain different constituency parsers with the help of modified training data to transform them into sentence segmenters. With these segmenters, documents are mapped to sequences of sentence-related "logical document structures" (cf. (Power et al., 2003)). The resulting segmenters aim to improve downstream tasks by providing additional structural information. In this context, we experiment with German dependency parsing. We show that for certain sentence categories, which can be determined automatically, improvements in German dependency parsing can be achieved using our segmenter for preprocessing. The assumption suggests that improvements in other languages and tasks can be achieved.

Keywords: sentence segmentation, logical document structure, constituency parsing, dependency parsing, CoNLL shared task

1. Introduction

Tools for sentence structure analysis, i.e. constituency and dependency parsers (Chen and Manning, 2014; Dyer et al., 2016a; Kiperwasser and Goldberg, 2016; Straka et al., 2016; Nguyen and Verspoor, 2018), operate on sentence units, the boundaries of which are recognized during preprocessing. However, current sentence boundary detectors (such as (Azzi et al., 2019; González-Gallardo and Torres-Moreno, 2018)) operate sequentially; they detect sequences of sentence beginnings and ends without their dependencies. In contrast to this, natural language sentences are often more deeply structured: they can, for example, consist of a main clause and several subordinate clauses or other segments (e.g. inserts in parentheses). They can even embed entire sentences recursively and then contain several sentence beginnings and ends. This happens, for example, when embedding direct speech into indirect speech, where embedded sentences can even be interrupted and continued later within the embedding sentence. This is illustrated by the following sentence from Kafka's *The Metamorphosis*, in which the sentence beginning with *Gregor* is interrupted, continued later, and followed by a second embedded sentence:

> "Gregor," it cried – it was the mother –, "it is three-quarter of seven. Weren't you going somewhere?"

Even under the assumption that a boundary detector recognizes this as a single sentence without terminating a supposed first sentence with the point after *seven*, it would not sufficiently recognize its inner structure, according to which a matrix sentence in indirect speech contains two sentences in direct speech. Our hypothesis is now that if parsers are better informed about the latter type of logical document structure (see (Power et al., 2003)), they can better analyze syntactic structures. In particular, we assume that a boundary detector that recognizes recursively structured sentence structures of this type is less error-prone and thus better prepares for subsequent parsing. To this end, we retrain constituency parsers to transform them into segmenters for detecting the latter kind of sentence structure. With these segmenters, documents are mapped to sequences of sentence-related "logical document structures" (cf. (Power et al., 2003)). We then use the segmenters to improve German dependency parsing. We show that for certain sentence categories, which can be determined automatically, improvements can be achieved using our segmenter for preprocessing. The assumption suggests that improvements in other languages and tasks can be achieved.

The paper is organized as follows: Section 2 gives an overview of related work, Section 3 describes the generation of training data for detecting sentence-related logical document structures. Section 4 deals with retraining constituency parsers for detecting these structures and Section 5 with using the resulting segmenters for the downstream task of dependency parsing. Section 6 contains our discussion and error analysis and Section 7 gives a conclusion and a prospect on future work.

2. Related Work

In the field of logical document structure analysis, there is already research in the form of *Sentence Boundary Detection* (SBD), where the task consists of recognizing the beginning and end of a sentence. For an approach to text classification that explores logical document structures, see (Mehler et al., 2007).

Attempts were made to solve the task of SBD using different approaches and methods for different languages, including supervised and unsupervised methods. Among the su-

pervised variants are maximum entropy (Reynar and Ratnaparkhi, 1997; Choi, 2000; Wang and Huang, 2003; Agarwal et al., 2005), rule-based (Wang and Huang, 2003), Markov Models (Jurish and Würzner, 2013), conditional random fields (Liu et al., 2005; Oba et al., 2006) and the ever increasingly popular neural networks (Xu et al., 2014; Treviso et al., 2017; González-Gallardo et al., 2018; Du et al., 2019; Schweter and Ahmed, 2019). Another class of methods for the SBD are the unsupervised methods (Kiss and Strunk, 2006; Strunk et al., 2006), which in contrast to the supervised variants do not require any additional training resources.

The methods presented here have in common that they operate sequentially, whereby only the beginning and end of a sentence are determined and thus it is assumed that sentences have no hierarchy. It is neglected that natural language sentences can be deeply nested by dependent clauses. In this paper we approach this desideratum by developing a tagger that recognizes recursively structured sentence structures. (Vogel and Fischer, 2019) retrain a *Recurrent Neural Network Grammar* (RNNG) to detect sentence-level related logical document structures. We follow this approach and also retrain an RNNG and add a comprehensive evaluation and error analysis.

3. Training Data

For detecting sentence-related logical document structures we require training data that already divides sentences into segments and displays these segments in a hierarchical relationship to each other. We could not use a corresponding gold standard corpus as a training database, as in classical dependency or constituency parsing, since there was no such resource for the language addressed in this paper, that is, German. Thus, we generated such a resource on our own. On the basis of 17 works by the German-speaking authors Franz Kafka and Thomas Mann, we first developed a training data set using a rule-based system that separates sentences into different sentence segments. This rule-based system basically operates as a counting system of candidate sentence and phrase boundary markers (whether pairing or not). In this way, it detects recursively nested (sub-)sentence structures and thereby explores sets of training data that can be used to train LDS detection systems. However, because of its rule-based nature, this system is not robust so that it basically stops processing when encountering heretofore unseen logical document structures or if the input document wrongly manifests these structures (e.g. by missing parentheses). As a consequence, the kind of machine learning presented here is indispensable in order to make the latter LDS annotation robust.

Each sentence segment of our training corpus is then mapped onto a hierarchical structure consisting of the root element (ROOT) as well as sentence (S), segment (SEG), word (W) and character (C) elements. The main record is the root element, to which possible subordinate clauses and other insertions (e.g. insertions in parentheses) are hierarchically subordinated. As a result, we get a training set consisting of 14,371 segmented sentences. The sentences are stored in a format corresponding to a modified form of the Penn Treebank (Marcus et al., 1993).

Parser	F1 Ø in %	Time (per sentence) Training	Parsing
PCFG (Stanford)	**69.33**	0.006 s	11.161 s
Shift-Red. (Stanford)	**82.68**	0.053 s	0.188 s
RNNG (Dyer et al.)	**87.16**	0.207 s	1.837 s

Table 2: F1-score and processing times of tested constituency parsers.

Author	Work	Sent.	Seg.	Tok.
F. Kafka	Auf der Galerie	2	54	288
F. Kafka	Blumfeld ein älterer Junggeselle	382	1 249	8 641
F. Kafka	Briefe an den Vater	672	2 744	16 275
F. Kafka	Das Schloss	5 837	21 915	109 025
F. Kafka	Das Urteil	284	684	3 993
F. Kafka	Der Bau	416	1 974	12 869
F. Kafka	Der Gruftwaechter	490	960	4 004
F. Kafka	Der Heizer	606	1 675	10 200
F. Kafka	Ein Fragment Der Heizer	5 599	15 905	89 139
F. Kafka	Die Verwandlung	832	3 099	19 151
F. Kafka	Ein Hungerkünstler	112	580	3 422
F. Kafka	In der Strafkolonie	634	1 879	10 259
T. Mann	Der Tod in Venedig	1 074	4 406	25 024
T. Mann	Der kleine Herr Friedemann	1 947	5 567	31 534
T. Mann	Gladius Dei	372	908	5 175
T. Mann	Königliche Hoheit	6 072	18 667	107 256
T. Mann	Schwere Stunde	160	545	2 797

Table 1: Used works in training

4. Re-Training constituency parsers

Our sentence segmenter essentially examines recursively embedded sentence segments, which are represented in hierarchical parse trees. Obviously, parsers for syntactic constituency structures fit this task. A central advantage of machine learning parsers is that they can be transferred (reuse) to related tasks with the help of new training data without having to change their source code. This approach is taken up in this paper. Classical constituency parsers as given by *Recurrent Neural Network Grammar*s (RNNG) (Dyer et al., 2016b) are trained with tree-like training data. Our newly created training corpus (see Section 3) also consists of tree-like structures. In order to obtain the desired segmenters for hierarchical sentence segment structures, it is thus essentially sufficient to retrain existing parsers. To this end, we experiment with three parsers: the (unlexicalized[1]) Stanford PCFG parser (Klein and Manning, 2003), the Stanford Shift Reduce parser (Zhu et al., 2013) and the RNNG parser (Dyer et al., 2016b) which was chosen as the most promising parser due to its F1-scores. In order to evaluate these parsers, we use the evaluation tool *evalb* (Sekine and Collins, 1997). Table 2 shows the corresponding results (training, validation and test sets were split according to the 60/20/20 rule): RNNG is obviously the best performing parser; but it is also most time-consuming in terms of training. The shift-reduce parser is the fastest in training and parsing, while RNNG is in the middle of the rating. From this point of view, RNNG is a top candidate for our task.

[1]We also tested the lexicalized model (Klein and Manning, 2002) but it performed worse.

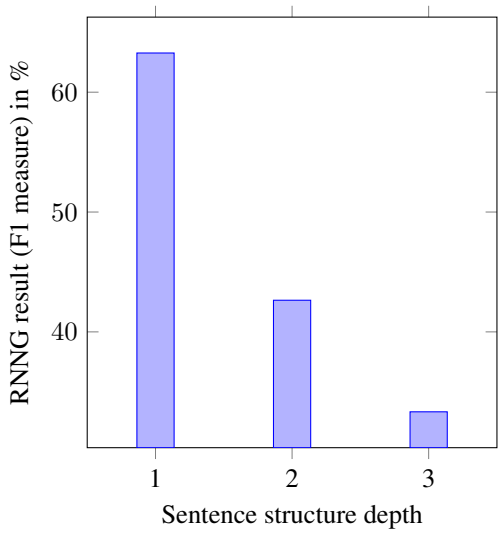

Figure 1: Result of RNNG-segmentation (F1-score) depending on sentence structure depth.

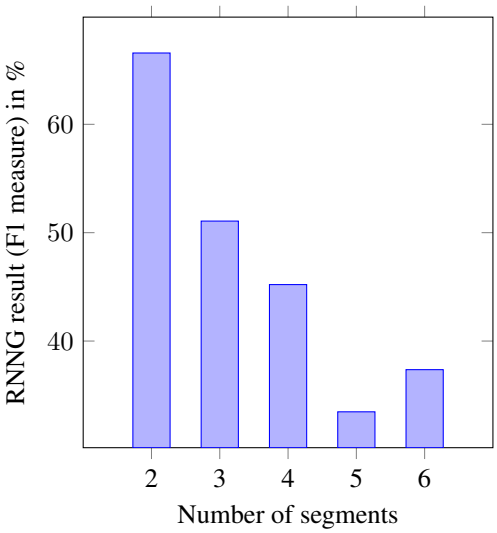

Figure 2: Result of RNNG-segmentation (F1-score) depending on the number of segments.

4.1. Error analysis

We briefly analyze for which sentences the segmentation using the RNNG parser is better or worse. The quality of the segmentation is crucial, as it has a central influence on whether improvements can be achieved in downstream tasks. The dependence of the segmentation result on the sentence structure depth (Figure 1) and on the number of segments (Figure 2) is particularly noticeable. This analysis shows that more complicated sentences are processed worse than simpler[2]. This observation was to be expected, since more complicated sentences, e.g. with a sentence structure depth of 3, occur less frequently in the training data than simple sentences, which becomes clear in Figure 3.[3]

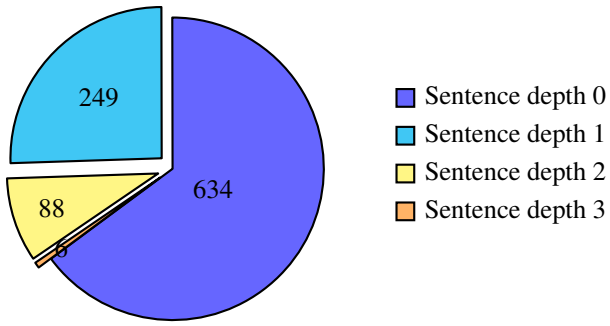

Figure 3: Allocation of the 977 CoNLL test sentences.

4.2. Differentiating authors through segment analysis

We additionally parsed several works of different authors using the RNNG-based segmenter. The resulting segmentations show similarities, but also differences between the authors. In Figure 4 the prose works (shown in Table 1) of the authors Franz Kafka and Thomas Mann and 6 works of Hermann Hesse and 4 works of Johann Wolfgang von Goethe are compared by the average number of segments manifested by them per sentence[4]. In Figure 5 we complement a comparison based on the average sentence structure depth. The figures show strong similarities in writing style between Kafka and Mann. Hesse and Goethe differ in the use of fewer (Hesse) or more segments (Goethe) and in the (Hesse) smaller or greater (Goethe) nesting of sentences.

5. Dependency Parsing with prior Segmentation

The segmentation applied here provides additional information about the structure of sentences. This information can now be used for downstream tasks such as dependency parsing. To this end, we take the CoNLL shared task on dependency parsing from 2018 (Zeman and Hajic, 2018) as an example to improve state-of-the-art parsers, trained with state-of-the-art training data within a current shared task. First we had to choose a suitable dependency parser, which is used in this shared task, whereby we decided for the TurkuNLP parser of (Kanerva et al., 2018). The reason for choosing this parser is its performance (LAS of 80.08%) and the simple implementation as a docker container. Building on this, we implemented the following parsing procedure: Initially, all input sentences were segmented by the RNNG-segmenter. The dependency parser then analyzed the sentence segments individually and reassembled the analysis results at the end. We pursued several strategies for the composition of the parsing results of the individual segments, which are presented in the following subsections.

[2]The only exception is the value for sentences with 6 segments in Figure 2, which, however, only applies to 2 sentences and is therefore not significant.

[3]Complex sentences are also processed worse because the number of parsing possibilities increases exponentially with the number of segments. The probability of finding exactly the right parse therefore decreases exponentially.

[4]For display reasons, statistical outliers of 40 segments per sentence or more had to be truncated, as values of up to 400 segments per sentence were detected due to incorrect segmentations.

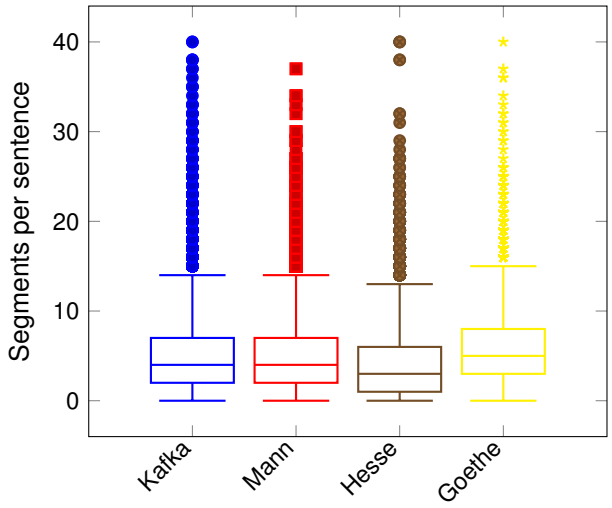

Figure 4: Comparing authors based on the number of segments per sentence.

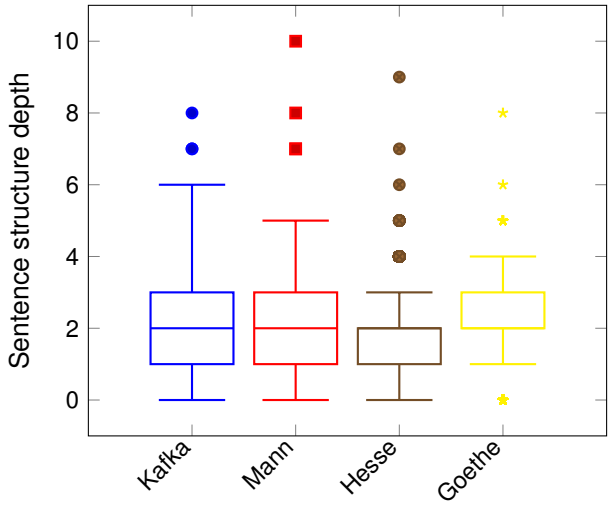

Figure 5: Comparing authors based on sentence structure depth.

5.1. General Procedure

The general procedure for splitting (segments) and assembling (parsing results) is carried out in three steps:

1. Segment the sentence
2. Parse and join segments individually (but do not link them yet!)
3. Link segment results to obtain a contiguous sentence.

Example: **Step 1:** The sentence *"After the house is renewed, I can move in."* is segmented as follows:

```
(S
  (SEG
    (W After) (W the) (W house) (W is) (W renewed) (C ,)
  )
  (W I) (W can) (W move) (W in) (C .)
)
```

Step 2: The resulting CoNLL rows of the subordinate clause are then inserted into the entire clause at the point where they appear in the original sentence (i.e. unsegmented sentence). The CoNLL-rows are only inserted correctly one below the other, but there is no link yet, that is, no change in the HEAD and DEPREL values. Thus the sentence appears correct at first, since all words follow one another in the correct order. However, the words only have relationships to words from their own segment (words in the main sentence are interdependent and words in the subordinate sentence are interdependent), but words of different segments are not linked as is expected for a correctly parsed sentence. In addition, commas removed before the segmentation are added again, but first with processing the head "?", since this head is only determined in the next step. According to the CoNNL-U format[5], the result of step 2 looks as follows:

ID	FORM	LEMMA	UPOS	HEAD	DEPREL
1	After	after	SCONJ	5	mark
2	the	the	DET	3	det
3	house	house	NOUN	5	nsubj:pass
4	is	is	AUX	5	aux:pass
5	renewed	renew	VERB	0	root
?	,	,	PUNCT	?	punct
1	I	I	PRON	3	nsubj
2	can	can	AUX	3	aux
3	move	move	VERB	0	root
4	in	in	ADV	3	advmod
5	.	.	PUNCT	3	punct

In the third step, the main and subordinate clauses must be connected to build a coherent sentence. For instance: If a subordinate clause is inserted in the middle of a main clause, the relationship of a word at the beginning of the main clause must be "extended" to a word at the end of the main clause, i.e. offset must be added to the head element which is as large as the number of words in the subordinate clause. That is, in a first sub-step, we store for each word which other word it referred to before the segmentation. Then, all IDs are numbered as in the original parse (in our example from 1 to 11). Question marks as values of the IDs of commas (see the example above) are overwritten and correctly set. As a result, all IDs are determined, but the dependencies do not necessarily match anymore. For example, the word *"can"* (ID: 2) is originally linked to *"move"* (ID: 3). After the IDs are renumbered from 1-11, the head of *"can"* has still the ID 3, but this ID now refers to *"house"*. Therefore, we store for each word in Step 2 to which other word it referred within its segment before the link. The IDs are then adjusted to the new values. In our example, *"can"* is linked to *"move"* again, which now has ID 9, so that we have to add the offset 6 to get the correct head ID ($3 + 6 = 9$). Finally, we get a sentence

[5]Here we have chosen a condensed format, which just shows ID, FORM, LEMMA, UPOS (universal PoS tag), HEAD and DEPREL (dependency relation). The CoNLL-U format also knows XPOS (language specific PoS tag), FEATS (morphological features), DEPS (enhanced dependencies) and MISC (miscellaneous), which we do not use and which also does not get measured by LAS. See https://universaldependencies.org/format.html.

with linked (main and subordinate) clauses, which is ready for evaluation:

ID	FORM	LEMMA	UPOS	HEAD	DEPREL
1	After	after	SCONJ	5	mark
2	the	the	DET	3	det
3	house	house	NOUN	5	nsubj:pass
4	is	is	AUX	5	aux:pass
5	renewed	renew	VERB	9	advcl
6	,	,	PUNCT	9	punct
7	I	I	PRON	9	nsubj
8	can	can	AUX	9	aux
9	move	move	VERB	0	root
10	in	in	ADV	9	advmod
11	.	.	PUNCT	9	punct

For some words, however, there are different strategies for how to link them with the rest of the sentence, which will be presented below.

5.2. Strategies for dealing with root elements

The final result must contain only one root element. For sentences with segments of different hierarchy levels, however, the segments are parsed individually, resulting in multiple root elements. The root element of the segment with the highest hierarchy level is always selected as the "main" root element. The other root elements must lose their root status and must be changed. Two different strategies have been developed for this purpose:

1. *SubRootsLikeOrgParseIfPossible:* The root elements of the subordinate clauses are linked to the entire clause as in the original parse, that is, in the parse before the segmentation, wherever possible. In some cases this leads to loops, that is, mutual or circular relations over 3 or more words that are not allowed. In these cases, the following strategy is chosen:

2. *SubRootToUpperRoot:* The head of a child root element is set to the root element of the parent segment. This strategy is guaranteed loop-free, but performed -3.02% worse than the first strategy.

5.3. Strategies for dealing with commas

Before the sentences are parsed, the commas are removed and when the segments are rejoined, they are inserted back into the sentence and structurally integrated. There are two strategies for this integration:

1. *AllCommaToMainRoot:* An analysis of randomly selected sentences showed that commas usually refer to the root element of the sentence. In the evaluation, this strategy achieved the best result.

2. *AllCommaLikeOrgParse:* This strategy is similar to the *AllCommaToMainRoot* strategy, because also in the original parse, most commas referred to the root element of the sentence as head. Nevertheless, this strategy performs worse (-0.09%).

Category	Strategy	Result
Root Elements	SubRootsLikeOrgParseIfPossible	Ref./±0
	SubRootToUpperRoot	-3.02%
Commas	AllCommaToMainRoot	Ref./±0
	AllCommaLikeOrgParse	-0.09%
Relative Pronouns	RelPronLikeOrgParse	Ref./±0
	ChangeNothing	-0.11%

Table 3: Results of segment assembling strategies

5.4. Strategies for dealing with relative pronouns

In the case of relative pronouns, a disadvantage of segmentation that can occur in German must be compensated. In German there are the relative pronouns *der*, *die*, *das*, which are all translated to *the* and which are identical in form to the German article. If a relative sentence is processed separately, it no longer "knows" that a token like *der* in the original sentence was a relative pronoun (though it should recognize this by examining the word order). The parser therefore recognizes it in rare cases as an article and not as a relative pronoun. To deal with this problem, we distinguish two strategies:

1. *RelPronLikeOrgParse:* Words that are relative pronouns in the original sentence are replaced by the values UPOS, XPOS and FEATS to the values of the original sentences. Note that only the above 3 values are changed, but not HEAD and DEPREL. The word type is therefore changed, but not the dependencies, since only the wrong detection of the article is to be avoided, while the sentence structure remains unchanged.

2. *ChangeNothing:* Here, the result of the segment-related parsing is adopted and the risk of incorrect recognition of a relative pronoun as an article is accepted, resulting in a deterioration of -0.11% compared with the first strategy.

Table 3 summarizes our evaluation of all strategies described so far.

6. Results and Error Analysis

All results of dependency parsing are presented in the following section. According to section 4 we selected the RNNG parser was preferred over the other two parsers. Therefore, only results obtained either by manual segmentation (called gold segmentation) or by segmentation using RNNG are presented.

6.1. Results

The manual segment analysis (gold segmentation) of the sentences from the CoNLL Shared Task showed that 351 out of 977 sentences have a hierarchical structure, so that they can be divided into a main and a subordinate clause, for example. Sentences that hypotactically link main clauses or consist of a single clause remain unchanged; they are treated as if they were not segmented.

Content	Original(%)	Gold Δin%	RNNG Δin%	Start word	Rate?
Das einzige was nervt ist, dass man sich sein Handtuch selbst mitbringen muss.	86.67	+06.66	**+06.66**	*dass*	Yes
"So sicher sind wir uns heute nicht mehr", sagt Braun.	78.57	+14.29	±00.00	*So*	No
Zwei Geiseln seien nach Angaben der Entführer leicht erkrankt, würden aber behandelt, erklärte ein Behördensprecher	83,33	±00.00	-22,22	*würden*	No

Table 4: Parsing results of two sentences (extract).

Table 4 shows two German[6][7][8] sample sentences which, according to the gold segmentation, have a hierarchical structure and, therefore, if the RNNG-based segmentation detects this hierarchy, can achieve different results as a result of our segmentation. The first column of Table 4 shows the sentences and the second column the original parsing result (LAS; without segmentation). The third column shows the result (delta compared to the original parsing result) obtained using the gold segmentation and the fourth column the result obtained using the RNNG-segmentation. The fifth column shows the start word of the subordinate clause. Now we made the following observation: If a sentence has a simple structure (e.g. consisting of a main and a subordinate clause) and if the start word is "*der*" (*the*), "*die*" (*the*), "*das*" (*the*), "*dass*" (*that*) or "*weil*" (*because*), the sentence belongs to a small group of sentences for which significant improvements in dependency parsing can be achieved by using the RNNG-segmenter. For these sentences, the last column "Rate?" in Table 4 is set to "Yes" and they are considered separately below. Gold segmentation could also improve sentences with other segment starts, but RNNG segmentation could on average only improve sentences with the latter start words because they are often included in training data and can therefore be learned well.
To compute an overall evaluation, we processed all sentences according to the form of Table 4. Table 5 summarizes our findings: the original parsing results (without segmentation) are displayed in the first line – it is used as a reference. This relates to 359 sentences[9] (i.e. all sentences which have a hierarchical structure) and the 69 special sentences for which the last column in Table 4 is set to "Yes". The third row in table 5 shows the parser results for gold segmentation, that is, the theoretically achievable best values in the event that the RNNG parser achieved an F1-score of 100.00%. As shown in Table 2, however, it only reaches an F1-score of 87.16%. The parser results obtained by means of the RNNG-segmentation are displayed in the last row. The value of +0.94% is particularly noteworthy, as actual improvements were achieved for the corresponding sentence class, whose members can be determined automatically. Therefore, we created a simple rule based predictor tool, which selects our desired group of sentences (here: sentence is separated by exactly one or two commas and the subordinate clauses starts with 'der", "die", "das", "dass" or "weil"). We do not adjust our tool to this specific group of sentences (which could cause a risk of overfitting), instead, we predict (rule-based), which have a chance of improving and will therefore be segmented by RNNG before they get dependency-parsed. All other sentences should be dependency-parsed without segmenting them before, as they will probably worsen the result.

Parsing method	Ø LAS change with regard to ...	
	processed sentences (359)	considered sentences (69)
Original	79.02% (reference)	80.57% (reference)
Gold	+0.95%	+2.26%
RNNG	-2.63%	**+0.94%**

Table 5: Average parsing results: original versus gold segmentation and RNNG-segmentation.

6.2. Error Analysis

The effect of sentence structure depth on parsing accuracy is shown in Figure 1. This depth also affects dependency parsing, that is, not only the segmenter but also the Turku parser has problems parsing deeply nested sentences. This dependency is illustrated in Figure 6. There is a second dependency between the original parsing results and the ones obtained by means of the gold and RNNG-segmentation, respectively, which is shown in Figure 7. On average, improvements by means of gold or RNNG-segmentations can only be achieved if the original result is below 80% (but since this is a mean value, there are counterexamples). If the original result is above 80%, it is rather difficult to improve the result further. This connection is obvious. If the original result is already good, it is more difficult to improve it – on the contrary, it is usually not changed or worsened, as the right side of the diagram shows. In summary, particularly complex sentences and sentences where

[6]English translation of the 1st sentence: The only thing that sucks is that you have to bring your own towel.

[7]English translation of the 2nd sentence: "We are not that sure of ourselves today," says Braun.

[8]English translation of the 3rd sentence: According to the kidnappers, two hostages were slightly ill, but were being treated, a spokesperson for the authorities explained.
Due to the more complex sentences structure, this is an example of a sentence, where the segmentation approach fails.

[9]The number 359 is composed of 351 sentences that have a hierarchical structure after gold segmentation and 8 sentences in which RNNG erroneously recognizes a hierarchical structure (false positives).

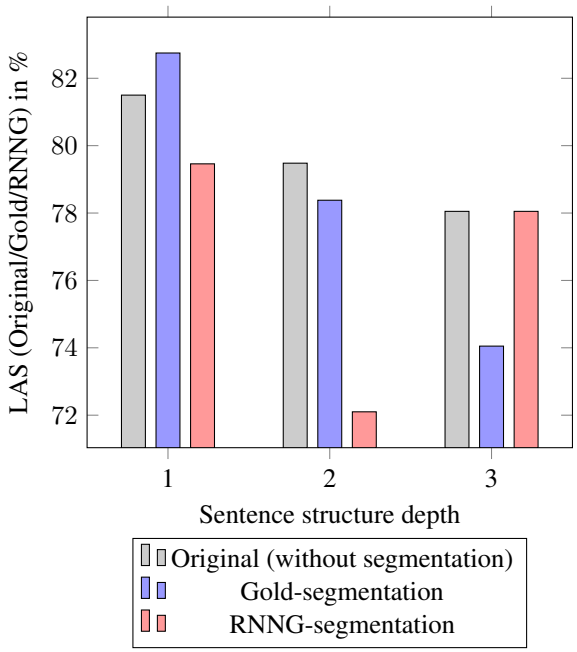

Figure 6: Parsing results (original, gold segmentation and RNNG-segmentation) (LAS) as a function of sentence depth.

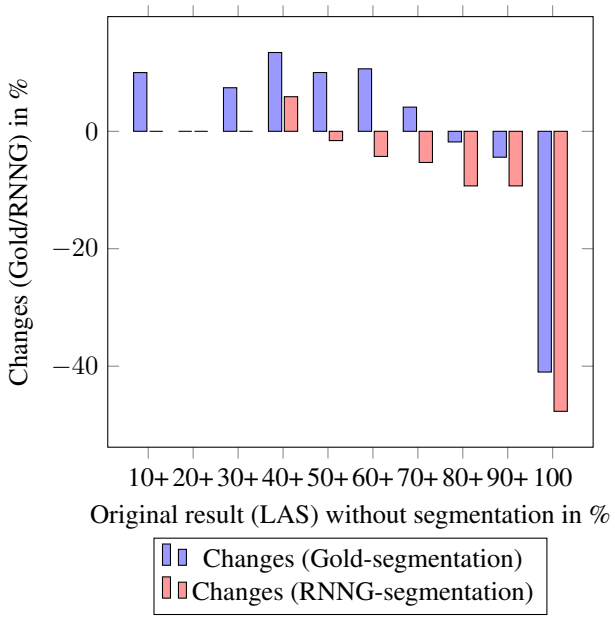

Figure 7: Changes induced by gold and RNNG-segmentation as a function of the original parsing result (LAS).

the original result is already good or very good are difficult to improve by segmentation. However, if the original parsing is worse and the sentence structure is simpler, then we find worthwhile candidates whose parsing can be improved by our approach.

7. Conclusion

We presented a segmenter for detecting sentence structures whose preprocessing can in certain cases help to improve dependency parsing. The segmenter was chosen in an experiment in which we retrained parsers of constituency structure by means of gold standard data that was created for detecting sentence structures on the level of clauses and related segments (i.e. sentence-related logical document structure). We selected the RNNG-based segmenter because it performed best and tested it as a preprocessor for dependency parsing based on the TurkuNLP parser. Though the segmentation-based preprocessing did not generally help to improve dependency parsing, we nevertheless identified a subclass of sentences for which it did. In this result, we recognize a potential for improving downstream tasks with the help of our segmentation approach. In addition, our segmenter recognizes much more complex structures than classic sentence boundary detectors. This concerns context-free structures of sentence segments that are overlooked by linearly operating detectors. Here, too, we see a potential for future work when it comes to determining the complexity of sentences more precisely than before. In future work we plan exactly this, the analysis of the sentence structure using the example of a large corpus of literature.

8. Open Source

The code of this paper and partly also the data is available on GitHub under https://github.com/texttechnologylab/LDS. Also, our web interface which implements the three different sentence segmenters is online via the TextImager https://textimager.hucompute.org/LDS/ (Hemati et al., 2016).

9. Bibliographical References

Agarwal, N., Ford, K. H., and Shneider, M. (2005). Sentence boundary detection using a maxent classifier. In *Proceedings of MISC*, pages 1–6.

Azzi, A. A., Bouamor, H., and Ferradans, S. (2019). The finsbd-2019 shared task: Sentence boundary detection in pdf noisy text in the financial domain. In *Proceedings of the First Workshop on Financial Technology and Natural Language Processing*, pages 74–80.

Chen, D. and Manning, C. D. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 740–750.

Choi, F. Y. Y. (2000). Advances in domain independent linear text segmentation. In *6th Applied Natural Language Processing Conference, ANLP 2000, Seattle, Washington, USA, April 29 - May 4, 2000*, pages 26–33.

Du, J., Huang, Y., and Moilanen, K. (2019). AIG Investments.AI at the FinSBD task: Sentence boundary detection through sequence labelling and BERT fine-tuning. In *Proceedings of the First Workshop on Financial Technology and Natural Language Processing*, pages 81–87, Macao, China.

Dyer, C., Kuncoro, A., Ballesteros, M., and Smith, N. A. (2016a). Recurrent neural network grammars. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016, pages 199–209.

Dyer, C., Kuncoro, A., Ballesteros, M., and Smith, N. A. (2016b). Recurrent neural network grammars. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 199–209.

González-Gallardo, C. and Torres-Moreno, J. (2018). Sentence boundary detection for french with subword-level information vectors and convolutional neural networks. *CoRR*, abs/1802.04559.

González-Gallardo, C., Pontes, E. L., Sadat, F., and Torres-Moreno, J. (2018). Automated sentence boundary detection in modern standard arabic transcripts using deep neural networks. In *Fourth International Conference On Arabic Computational Linguistics, ACLING 2018, November 17-19, 2018, Dubai, United Arab Emirates*, pages 339–346.

Hemati, W., Uslu, T., and Mehler, A. (2016). TextImager: a distributed UIMA-based system for NLP. In *Proc. of COLING 2016: System Demonstrations*, pages 59–63.

Jurish, B. and Würzner, K. (2013). Word and sentence tokenization with hidden markov models. *JLCL*, 28(2):61–83.

Kanerva, J., Ginter, F., Miekka, N., Leino, A., and Salakoski, T. (2018). Turku neural parser pipeline: An end-to-end system for the conll 2018 shared task. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Brussels, Belgium, October 31 - November 1, 2018*, pages 133–142.

Kiperwasser, E. and Goldberg, Y. (2016). Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL*, 4:313–327.

Kiss, T. and Strunk, J. (2006). Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.

Klein, D. and Manning, C. D. (2002). Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 3–10.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 7-12 July 2003, Sapporo Convention Center, Sapporo, Japan.*, pages 423–430.

Liu, Y., Stolcke, A., Shriberg, E., and Harper, M. P. (2005). Using conditional random fields for sentence boundary detection in speech. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 451–458.

Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1993). Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Mehler, A., Geibel, P., and Pustylnikov, O. (2007). Structural classifiers of text types: Towards a novel model of text representation. *Journal for Language Technology and Computational Linguistics (JLCL)*, 22(2):51–66.

Nguyen, D. Q. and Verspoor, K. (2018). An improved neural network model for joint POS tagging and dependency parsing. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Brussels, Belgium, October 31 - November 1, 2018*, pages 81–91.

Oba, T., Hori, T., and Nakamura, A. (2006). Sentence boundary detection using sequential dependency analysis combined with crf-based chunking. In *INTERSPEECH 2006 - ICSLP, Ninth International Conference on Spoken Language Processing, Pittsburgh, PA, USA, September 17-21, 2006*.

Power, R., Scott, D., and Bouayad-Agha, N. (2003). Document structure. *Computational Linguistics*, 29(2):211–260.

Reynar, J. C. and Ratnaparkhi, A. (1997). A maximum entropy approach to identifying sentence boundaries. In *5th Applied Natural Language Processing Conference, ANLP 1997, Marriott Hotel, Washington, USA, March 31 - April 3, 1997*, pages 16–19.

Schweter, S. and Ahmed, S. (2019). Deep-eos: General-purpose neural networks for sentence boundary detection. In *Proceedings of the 15th Conference on Natural Language Processing, KONVENS 2019, Erlangen, Germany, October 9-11, 2019*.

Sekine, S. and Collins, M. (1997). Evalb bracket scoring program. *http://www.cs.nyu.edu/cs/projects/proteus/evalb*.

Straka, M., Hajic, J., and Straková, J. (2016). Udpipe: Trainable pipeline for processing conll-u files performing tokenization, morphological analysis, POS tagging and parsing. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*.

Strunk, J., Jr., C. N. S., and Kaestner, C. A. A. (2006). A comparative evaluation of a new unsupervised sentence boundary detection approach on documents in english and portuguese. In *Computational Linguistics and Intelligent Text Processing, 7th International Conference, CICLing 2006, Mexico City, Mexico, February 19-25, 2006, Proceedings*, pages 132–143.

Treviso, M. V., Shulby, C., and Aluísio, S. M. (2017). Sentence segmentation in narrative transcripts from neuropsychological tests using recurrent convolutional neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 315–325.

Vogel, F. and Fischer, P. (2019). Practical course report of the practical course textimaging at goethe university frankfurt, institute for informatics. unpublished.

Wang, H. and Huang, Y. (2003). Bondec–a sentence boundary detector. *CS224N Project, Stanford*.

Xu, C., Xie, L., Huang, G., Xiao, X., Chng, E., and Li, H. (2014). A deep neural network approach for sen-

tence boundary detection in broadcast news. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 2887–2891.

Daniel Zeman et al., editors. (2018). *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Brussels, Belgium, October 31 - November 1, 2018*. Association for Computational Linguistics.

Zhu, M., Zhang, Y., Chen, W., Zhang, M., and Zhu, J. (2013). Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 434–443.