Article

Not peer-reviewed version

Hierarchical Clause Annotation: Building a Clause-Level Corpus for Semantic Parsing with Complex Sentences

Yunlong Fan , Bin Li , Yikemaiti Sataer , Miao Gao , Chuanqi Shi , Siyi Cao , Zhiqiang Gao *

Posted Date: 7 June 2023

doi: 10.20944/preprints202306.0530.v1

Keywords: Clause Hierarchy; Hierarchical Clause Annotation; Complex sentences; Semantic Parsing; Syntactic Parsing; RST corpus

Article

Hierarchical Clause Annotation: Building a Clause-Level Corpus for Semantic Parsing with Complex Sentences

Yunlong Fan [1,2], Bin Li [1,2], Yikemaiti Sataer [1,2], Miao Gao [1,2], Chuanqi Shi [1,2], Siyi Cao [3] and Zhiqiang Gao [1,2,*]

1 School of Computer Science and Engineering, Southeast University, Nanjing 211189, China; fanyunlong@seu.edu.cn (Y.F.); lib@seu.edu.cn (B.L.); yikmat@seu.edu.cn (Y.S.); miaogao@seu.edu.cn (M.G.); chuanqi_shi@seu.edu.cn (C.S.)

2 Key Laboratory of Computer Network and Information Integration, Southeast University, Ministry of Education, Nanjing 211189, China

3 School of Foreign Languages, Southeast Univesity, Nanjing 211189, China; siyic@seu.edu.cn (S.C.)

* Correspondence: zqgao.seu.edu.cn (Z.G.)

Featured Application: Hierarchical clause annotation could be applied in many downstream tasks of natural language processing, including abstract meaning representation parsing, semantic dependency parsing, text summarization, argument mining, information extraction, question answering, machine translation, etc.

Abstract: Most natural language processing (NLP) tasks suffer performance degradation when encountering long complex sentences, such as semantic parsing, syntactic parsing, machine translation, and text summarization. Previous works address the issue with an intuition of decomposing complex sentences and linking simple ones, such as RST-style discourse parsing, split-and-rephrase (SPRP), text simplification (TS), simple-sentence-decomposition (SSD), etc. However, these works are not applicable for semantic parsing like abstract meaning representation (AMR) parsing and semantic dependency parsing due to misalignments to semantic relations and unavailabilities to preserve original semantics. Following the same intuition and avoiding deficiencies of previous works, we propose a novel framework, hierarchical clause annotation (HCA), based on the linguistic research of clause hierarchy. With the HCA framework, we annotate a large HCA corpus to explore the potentialities of integrating HCA structural features into semantic parsing with complex sentences. Moreover, we decompose HCA into two subtasks, i.e., clause segmentation and clause parsing, and provide neural baseline models for more silver annotations.

Keywords: clause hierarchy; hierarchical clause annotation; complex sentences; syntactic parsing; semantic parsing; RST corpus

1. Introduction

Most natural language processing (NLP) tasks suffer performance degradation when encountering long complex sentences, such as abstract meaning representation (AMR) parsing [1], semantic dependency parsing [2], constituency parsing [3], semantic role labeling [4], machine translation [5], and text summarization [6]. An intuition to address this issue is first to decompose complex sentences and then re-link simple ones, which shares similar ideas with tasks like RST-style discourse parsing (hereafter RST parsing) [7], split-and-rephrase (SPRP) [8], text simplification (TS) [9], simple-sentence-decomposition (SSD[1]) [10], etc.

[1] Gao et al. have yet to name their task, and thus we summarize their main idea into *simple-sentence-decomposition (SSD)* for the convenience of writing.

However, previous works with such intuitions to process complex sentences are not practical for semantic parsing tasks, such as AMR [11] and semantic dependency parsing [12]. RST parsing, aiming to extract the rhetorical relations among elementary discourse units (EDUs) [13] at a document level, is still an open problem, where the state-of-the-art model only achieves 55.4 and 80.4 *Parseval-Full* scores for multi- and intra-sentential parsing. Besides, the blurry definitions of EDUs and the misalignments between rhetorical relations and semantic relations make RST parsing unsuitable for semantic parsing. The SPRP task keeps a big splitting granularity, where outputs may still be complex sentences. The TS and SSD tasks, which decompose complex sentences into simple ones, can not preserve the original semantics for rephrasing for simpler syntax (TS) or dropping discourse connectives (SSD).

In this paper, we propose a novel task, hierarchical clause annotation (HCA), based on the linguistic research of clause hierarchy [14], where clauses are fundamental text units centering on a verb phrase, and sentences with multiple clauses form a complex hierarchy. Our HCA is a more lightweight task at the sentence level, has explicit definitions of clauses and appropriate mappings between inter-clause and semantic relations (vs. RST parsing), and aims to annotate complex sentences into a clause hierarchy (vs. SPRP) without changing or dropping any semantics (vs. TS and SSD).

To show the potentialities of HCA to facilitate semantic parsing with complex sentences, we demonstrate the HCA tree, AMR graph, and semantic dependency graph (SDG) of a complex sentence from the AMR 2.0 dataset[2] in Figure 1:

$$[If\ I\ do\ not\ check,]_{C_1}\ [I\ get\ very\ anxious,]_{C_2}\ [which\ does\ sort\ of\ go\ away\ after\ 15\text{-}30\ mins,]_{C_3}\ [but\ often\ the\ anxiety\ is\ so\ much]_{C_4}\ [that\ I\ can\ not\ wait\ that\ long.]_{C_5}$$

The above sentence is segmented into five clauses C_i, where

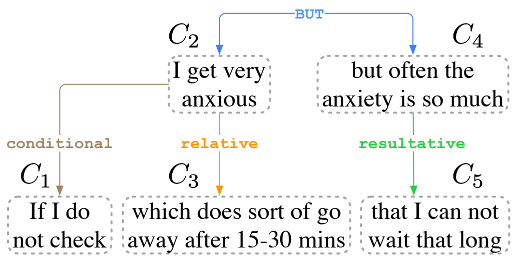
- C_2 and C_4 are coordinate and contrastive,
- C_1 and C_3 are conditional adverbial and relative clauses of C_2, respectively,
- C_5 is a resultative adverbial clause of C_4.

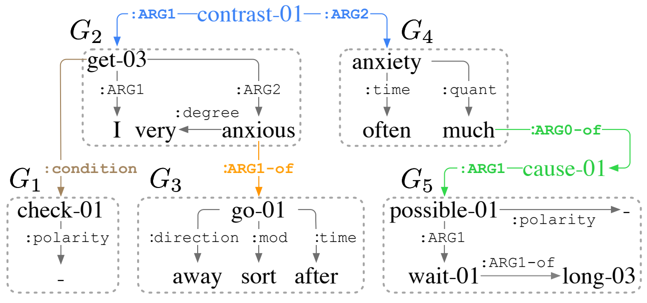
In the HCA tree, the coordinate relation *But* and clauses C_i are nodes, and subordinate relations are directed edges. As demonstrated in Figure 1, the HCA tree shares the same hierarchy with two semantic parsing representations, indicating the possibility of incorporating HCA's structural information into semantic parsing with complex sentences.

Our main contributions are as follows:

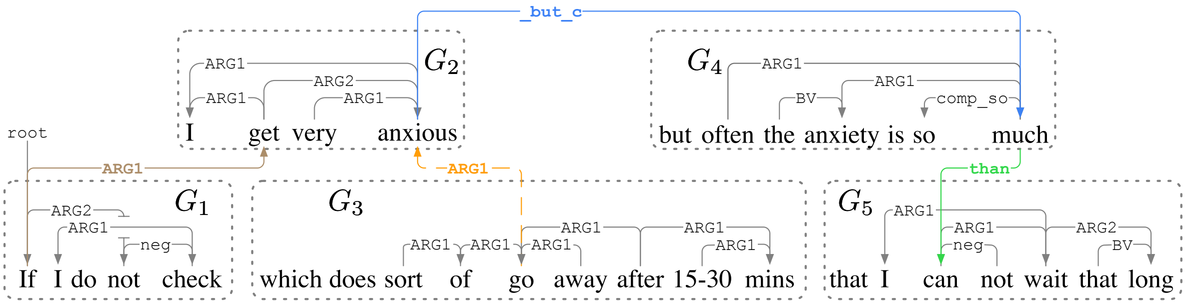
1. We propose a novel framework, hierarchical clause annotation (HCA), to segment complex sentences into clauses and capture their interrelations based on the linguistic research of clause hierarchy, aiming to provide clause-level structural features to facilitate semantic parsing tasks.
2. We elaborate on our experience developing a large HCA corpus – including determining an annotation framework, creating a silver-to-gold manual annotation tool, and ensuring annotating quality. The resulting HCA corpus contains $19,376$ English sentences from AMR 2.0, each including at least two clauses.
3. We decompose HCA into two subtasks, i.e., clause segmentation and parsing, and adapt discourse segmentation and parsing models for the HCA subtasks. Experimental results show that the adapted models achieve satisfactory performances in providing reliable silver HCA data.

[2] https://catalog.ldc.upenn.edu/LDC2017T10


BUT
C_2
I get very anxious
C_4
but often the anxiety is so much
conditional
relative
resultative
C_1
If I do not check
C_3
which does sort of go away after 15-30 mins
C_5
that I can not wait that long
(a) Hierarchical Clause Annotation Tree



(b) Abstract Meaning Representation Graph



(c) Semantic Dependency Graph (DM annotation)

Figure 1. Clauses C_i in (a) correspond to subgraphs G_i in (b) and (c), respectively. Colored directed edges in (a) are inter-clause relations, mapping the same colored AMR nodes and edges in (b) and semantic dependencies in (c). Note that reentrant AMR relations in (b) introduced by the pronoun "I" are omitted to save space, as well as semantic dependencies between orphan tokens and the root token "If" in (c).

2. Related Work

2.1. RST Parsing

In a document, the clauses, sentences, and paragraphs are logically connected together to form a coherent discourse. Rhetorical Structure Theory (RST) [13] provides a general way to describe the relations among parts in a text and postulates a hierarchical discourse structure called discourse tree (DT). The leaves of a DT can be a clause or a phrase without strict definitions, known as elementary discourse units (EDUs). Adjacent EDUs and higher-order spans are non-overlapping and connected hierarchically through coherence relations. Thus, coherence in discourse can be analyzed in terms of how a nucleus interacts with its surrounding satellites to communicate the relationships between main ideas and related ideas.

The RST parsing task generally requires breaking the text into EDUs (i.e., the discourse segmentation task) and linking the EDUs into a DT (i.e., the discourse parsing task). For discourse segmentation, Gessler et al. [15] proposed a Transformer-based neural classifier that enhances contextualized word embeddings with hand-crafted features and achieved the current state-of-the-art performance in DISRPT 2021 Shared Task on Discourse Unit Segmentation[3]. For discourse parsing, Kobayashi et al. [16] explored a strong baseline by integrating previous simple parsing strategies, top-down and bottom-up, with various transformer-based pretrained language models (PLMs).

Table 1 demonstrates the comparison between our HCA and RST parsing with exemplified sentences from RST Discourse Tagging Reference Manual[4]. Although both tasks aim to extract a tree structure from input texts, their definitions of elementary units and target interrelations vary, leading to the following differences.

[3] https://sites.google.com/georgetown.edu/disrpt2021?pli=1
[4] https://www.isi.edu/~marcu/discourse/

- The elementary units of HCA are clauses, while that of RST parsing are EDUs, including clauses and phrases. The blurry definitions of an EDU may cause obstacles in RST parsing. For example, phrases*"as a result of margin calls"* in (1) and *"Despite some their considerable incomes and assets"* in (2) are segmented as EDUs but not clauses due to the absence of a verb. Moreover, although the clause "that they have made it" functions as a predicative of the verb "feel", it can not be annotated as an EDU.
- The rhetorical relations in RST parsing characterize the coherence among EDUs, and some can not map to semantic relations. For example, the semantic relation between *"feel"* and *"that they have made it"* in (2) is not captured in RST parsing.

In summary, the blurry definitions of EDUs and the misalignments between rhetorical and semantic relations make RST parsing unsuitable for semantic parsing compared with our HCA.

Table 1. Comparison between our HCA task and the RST parsing task. Two exemplified sentences are from RST Discourse Tagging Reference Manual. Units (i.e., clauses or EDUs) are segmented by square brackets, and index marks $_i$. Relations between units are represented as arrows directed from a matrix clause or a nucleus EDU to a subordinate clause or a satellite EDU with a specific relation.

Task	**Output Descrpition**	**Output Example**	
		Unit	**Relation**
Hierarchical Clause Annotation	Clause trees built up by clauses and inter-clause relations	(1) [But some big brokerage firms said]$_1$ [they don't expect major problems as a result of margin calls.]$_2$	$1 \xrightarrow{objective} 2$
		(2) [Despite their considerable incomes and assets, one-fourth don't feel]$_1$ [that they have made it.]$_2$	$1 \xrightarrow{predicative} 2$
RST Parsing	Discourse trees built up by EDUs and rhetorical relations	(1) [But some big brokerage firms said]$_1$ [they don't expect major problems]$_2$ [as a result of margin calls.]$_3$	$1 \xrightarrow{attribution} 2$ $2 \xrightarrow{result} 3$
		(2) [Despite their considerable incomes and assets,]$_1$ [one-fourth don't feel that they have made it.]$_2$	$2 \xrightarrow{concession} 1$

2.2. Other Similar Tasks

Some similar tasks share the idea of decomposing complex sentences into simpler parts without capturing their interrelations.

2.2.1. Clause Identification

The CoNLL-2001 shared task, clause identification, [17] proposes a dataset with the gold standard clause provided by the Penn Treebank II [18], where clause-level tags (i.e., `S`, `SBAR`, `SBARQ`, `INV`, and `SQ`) indicate target clauses and clausal conjunctions. The clauses identified in the shared task comprise tensed clauses, non-tensed verb phrases, coordinators, and subordinators.

2.2.2. Split-and-Rephrase

The split-and-rephrase (SPRP) task [8] aims to split a complex input sentence into shorter sentences while preserving meaning. In that task, the emphasis is on sentence splitting and rephrasing. There is no deletion and no lexical or phrasal simplification, but the systems must learn to split complex sentences into shorter ones and to make the syntactic transformations required by the split(e.g., turn a relative clause into a main clause).

2.2.3. Text Simplification

The text simplification (TS) task [19] is the process of reducing the linguistic complexity of a text to improve its understandability and readability while maintaining its original information content and meaning. Typically, it rephrases complex sentences with simpler vocabulary and syntax and ignores trivial clauses from the source.

2.2.4. Simple-Sentence-Decomposition

The simple-sentence-decomposition (SSD) task [10] converts complex sentences into a covering set of simple sentences derived from the tensed clauses in the source sentence, where shared nouns or pronouns are copied, and discourse connectives (e.g., *and*, *but*, *although*, etc) are dropped.

2.2.5. Summary

Table 2 compares our HCA task and similar tasks above with the exemplified sentence in Section 1.

- For clause identification, coordinator "*but*" and subordinators "*If*", "*which*", and "*that*" are segmented out. Besides, non-tensed verb phrases that function as a subject, object, or postmodifier are also target clauses in the task. These cases are out of the definition of annotated clauses in our HCA framework, as redundant hierarchies occur in capturing inter-clause relations.
- For split-and-rephrase, the granularity of decomposing is larger than clauses due to the consideration of preserving the original meaning of the input sentence. The output (1) and (3) are still complex sentences with two clauses. Additionally, the output (2) is segmented from the relative clause that modifies the "anxious" in the matrix clause, leading to a syntax transformation.
- For text simplification, dropping the subordinator "If" and the coordinator "but" leads to the uncertainties of discourse relations between output sentences. Moreover, as replacements for simpler syntax in (3) and (4) bring misalignments between substitute and substituted tokens, text simplification is unsuitable to serve as a preprocess for semantic dependency parsing, which is a token-level task.
- For simple-sentence-decomposition, it also drops clausal connectives like text simplification, leading to the uncertainties of some discourse relations captured by semantic parsing.

Table 2. Comparison between our HCA task and similar tasks that decompose complex sentences into parts. The input sentence "*If I do not check, I get very anxious, which does sort of go away after 15-30 mins, but the anxiety is so much that I can not wait that long.*" is selected from the AMR 2.0 dataset and exemplified in Section 1. Underlined words in the **Output Example** column of each task are modified from the original sentence, while ~~crossed~~ words are deleted from the original sentence.

Task	Output Descrpition	Output Example
Hierarchical Clause Annotation	Finite clauses and non-finite clauses seperated by a comma	(1) If I do not check, (2) I get very anxious, (3) which does sort of go away after 15-30 mins, (4) but often the anxiety is so much (5) that I can not wait that long.
Clause Indentification	Finite clauses, non-tensed verb phrases, coordinators, and subordinators	(1) If (2) I do not check, (3) I get very anxious, (4) which (5) does sort of go away after 15-30 mins, (6) but (7) often the anxiety is so much (8) that (9) I can not wait that long.
Split-and -Rephrase	Shorter sentences	(1) If I do not check, I get very anxious. (2) The anxieties does sort of go away after 15-30 mins. (3) But often the anxiety is so much that I can not wait that long.

Table 2. *Cont.*

Task	Output Descrpition	Output Example
Text Simplification	Sentences with simpler syntax	(1) ~~If~~ I do not check. (2) I get very anxious (3) The anxiety lasts for 15-30 mins. (3) ~~But~~ I am often too anxious to wait that long.
Simple -Sentence -Decomposition	Simple sentences with only one clause	(1) ~~If~~ I do not check. (2) I get very anxious, (3) The anxiety does sort of go away after 15-30 mins. (4) ~~But~~ Often the anxiety is so much. (5) ~~that~~ I can not wait that long.

2.3. *Clause Hierarchy*

Clause Hierarchy can be described as a cline along which clauses distribute according to their different levels of grammatical integration [14,20–24]. These works propose that clause combinations in many languages can be described as a set of **tighter** or **looser** clauses. A tight clause means that a clausal constituent has, in comparison to a loose clause, more dependence on the clause with which it combines, typically a main clause. Table 3 shows three main versions of Clause Hierarchy and ordered linguistic phenomena according to their clause integration tightness degree.

Table 3. Three main types of clause hierarchy and the clines of their clause integration tightness degree.

Type	Cline of Clause Integration Tightness Degree
Matthiessen [21]	Embedded > Hypotaxis > Parataxis > Cohesive devices > Coherence
Hopper& Traugott [22]	Subordination > Hypotaxis > Parataxis
Payne [14]	Compound verb > Clausal argument > Relative > Adverbial > Coordinate > Sentence

Matthiessen [21] presented a type of clause hierarchy that extends from syntactic clause combination to cohesion and coherence at the discourse and text level. He considers clause combination to range from tight syntactic "embedding" (e.g., infinitival clauses as a complement to a main verb) to the looser relations of "hypotaxis" (e. g. a finite adverbial) to "parataxis" (e.g., coordination).

Hopper and Traugott [22] also offered a model that shares much with Matthiessen's, where "parataxis" is the syntactic independence of clauses and "hypotaxis" is a more integrated clause that is syntactically dependent within another clause's predicate. Tighter still is "subordination", which, like "embedding" in Matthiessen's type, covers all clauses which function as a constituent essential to grammaticality, e.g., verbal arguments.

Payne's clause hierarchy [14] extended from "compound verbs" (tightest) through to separate "sentences" (loosest), where clause combination becomes more or less a single verbal element in his type. Different from the above two types, Payne argues that compound verbs (e.g., *go get the book*), though uncommon in English, are considered the tightest clause combination as they have two verbal elements placed adjacently in a verb phrase, one of which lacks full finiteness.

In summary, the linguistic researches of Clause Hierarchy provide solid theoretical support for our HCA framework.

3. Hierarchical Clause Annotation Framework

To present an framework for annotation the clause hierarchy of complex sentences, we reference and modify Payne's version [14] of Clause Hierarchy due to his pellucid and comprehensive definitions of clause combination cases. As demonstrated in Figure 2, we do not consider *compound verbs* as a clause combination, as these cases are uncommon and produce one-verb clauses after annotation.

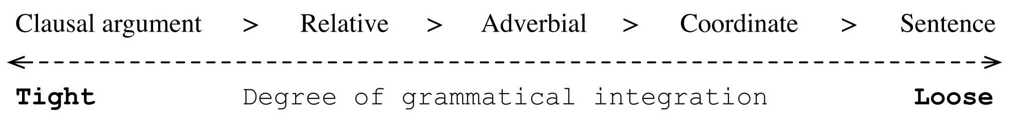

Clausal argument > Relative > Adverbial > Coordinate > Sentence
Tight
Degree of grammatical integration
Loose

Figure 2. Modified version of Payne's Clause Hierarchy.

With the above version of Clause Hierarchy, we synthesize the HCA framework and build a dataset under the guidance of the framework. The annotation work consists of a preprocessing stage with silver annotations transformed from existing schemas (constituent parsing and syntactic dependency parsing) and a manual proofreading phase with gold annotations on an elaborate browser-based annotator.

3.1. Concept

We list major concepts in the HCA framework.

3.1.1. Sentence and Clause

Sentences, typically starting with a capitalized word and ending with a complete stop, are principally units of written grammar and annotation inputs in HCA. A sentence must consist of at least one clause.

Clauses, considered core units of grammar, center around a verb phrase that largely determines what else must or may occur [25]. Clauses can be categorized by the inner verb type:

- Finite: clauses that contain tensed verbs;
- Non-finite : clauses that only contain non-tensed verbs like *ing*-participles, *ed*-participles, and *to*-infinitives.

In the HCA framework, finite clauses should be annotated, while non-finite clauses that are separated by a comma are also segmented out.

3.1.2. Clause Combination

The main ways in which clauses combine to form sentences are by joining clauses of equal syntactic status (coordination) and subordinate relation (subordination).

(1) Coordination and Coordinator

Coordination is an interrelation between clauses that share the same syntactic status and are typically connected by a coordinator such as *and*, *or*, *but*, etc. In addition, coordinators can be correlative structures (e.g., *either...or...* and *not only...but also...*) or just substituted by comma punctuation.

(2) Subordination, Subordinator, and Antecedent

Subordination occurs in a subordinate clause and a matrix clause that is superordinate to the subordinate clause. Subordination can be cataloged as follows:

- *Nominative:* Function as clausal arguments or noun phrases in the matrix clause and can be subdivided into *Subjective*, *Objective*, *Predicative* and *Appositive*.
- *Relative:* Define or describe a preceding noun head in the matrix clause.
- *Adverbial:* Function as a *Condition*, *Concession*, *Reason*, and such for the matrix clause.

Subordinators are the words that introduce a subordinate clause and indicate a semantic relation between the subordinate clause and its matrix clause, including subordinate conjunctions, relative pronouns, and relative adverbs. Simple subordinators contain a single word, e.g., *that*, *wh*-words, *if*, etc., while complex ones consist of more than one word, e.g., *as if*, *so that*, *even though*, etc.

Antecedents are nouns or pronouns modified before relative clauses and nouns explained before appositive clauses.

To better explain these HCA definitions, we demonstrate some example sentences which are segmented into multiple clauses in Table 4.

Table 4. Examples of sentences with different types of inter-clause relations. Clauses are segmented by square brackets and clause marks C_i. The underlined, double-underlined, and wave-underlined words are coordinators, subordinators, and antecedents, respectively.

	Relation	Example Sentence
		Coordination
(1)	*And*	[He should have been here at five]$_{C_1}$ [and he's not here yet.]$_{C_2}$
		Subordination
(2)	*Subjective*	[What we follow]$_{C_1}$ [is a foreign security strategic philosophy.]$_{C_2}$
(3)	*Objective*	[He knows]$_{C_1}$ [what it takes to start a business here.]$_{C_2}$
(4)	*Predicative*	[The reason is]$_{C_1}$ [that you lack confidence.]$_{C_2}$
(5)	*Appositive*	[I've accepted defeat]$_{C_1}$ [that this year of my life is a failure.]$_{C_2}$
(6)	*Relative*	[We've entered into an age]$_{C_1}$ [when dreams can be achieved.]$_{C_2}$
(7)	*Adverbial*	[He'd need to do his exam]$_{C_1}$ [before he went.]$_{C_2}$

3.2. HCA Representation

As illustrated in Figure 3, we model the two basic hierarchical schemas with concepts defined in Section 3.1, characterizing inter-clause relations with the same *nucleus-satellite* pattern in RST. To be specific, coordination is a multinuclear relation that involves two or more clauses (denoted as *nucleus* node C_i) dominated by the coordination node *co*, while subordination is a mononuclear relation (denoted as a directed edge *sub*) pointing from the matrix clause C_1 (*nucleus*) to its subordinate clause C_2 (*satellite*).

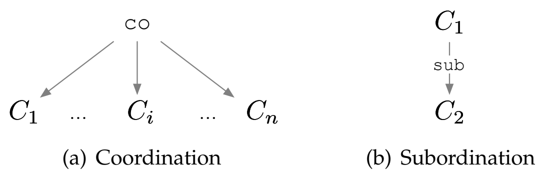


Figure 3. Two basic hierarchical schemas in HCA, where node C_i, node *co*, and edge *sub* represent a clause, coordination and subordination, respectively.

As a sentence consists of more clauses, its HCA representation can be a tree structure, where each node is a clause or an inter-clause coordination, and each directed edge is an inter-clause subordination. A three-layer HCA tree of a complex sentence involving five clauses and four interrelations is demonstrated in Figure 1(a).

3.3. HCA corpus

With the annotation framework discussed above, we aim to build an HCA corpus for further research on the possibilities of applying clausal structure features to semantic parsing tasks. We choose the AMR 2.0 dataset as our corpus base, whose 39,260 sentences are collected from the DARPA BOLT and DEFT programs, various newswire data, and weblog data.

The annotation work is conducted in two phases. First, two existing syntactic features, i.e., constituent and syntactic dependency parse trees, are employed to produce silver HCA annotations with transformation rules. Second, human annotators with prior English grammar research experience and extensive hands-on annotation training, review and modify silver annotations on a browser-based annotation tool.

3.3.1. Silver Data from Existing Schemas

Previous researchers[26–29] utilize constituent-based and syntactic dependency parse trees to extract clauses from sentences with some manual rules. Following the experience from these works, we employ Stanza [30] as our constituent parser and syntactic dependency parser to obtain silver HCA data.

- Consituency Parse Tree

The constituency parse tree (CPT) represents the syntactic structure of a sentence using a tree, where the nodes are sub-phrases that belong to a specific category in the grammar, and the edges are unlabeled. The transformation from CPT to the silver HCA data consists of three phases:

1. Traverse non-leaf nodes in the CPT and find the clause-type nodes: `S`, `SBAR`, `SBARQ`, `INV`, and `SQ`.
2. Identify the tokens dominated by a clause-type node as a clause.
3. When a clause-type node dominates another one, an inter-clause relation between them is determined without an exact relation type.

As demonstrated in Figure 4, the first two clauses of the sentence exemplified in Section 1 are identified through their constituent parse tree. The `SBAR` node and its child node `S` are combined as a single clause, as no `VP` is dominated by the other child constituent `IN` of `SBAR`. Moreover, the `S` node on the top dominates the `SBAR` node, indicating subordination between the two clauses in dashed boxes.

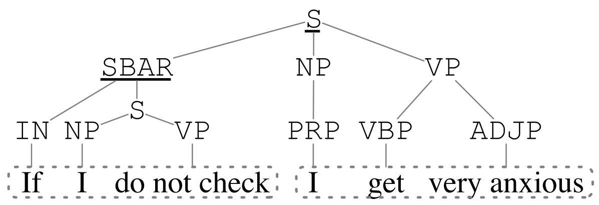


Figure 4. Extract clauses and inter-clause relations via the constituency parse tree. Two clauses in dashed boxes are identified by underlined clause-type nodes `S` and its child node `SBAR`. Note that child constituent nodes of the left `VP` and the right `ADJP` are omitted to save space.

- Syntactic Dependency Parse Tree

The syntactic dependency parse tree (SDPT) consists of a set of directed syntactic relations between the words in the sentence whose root is either a non-copular verb or the subject complement of a copular verb. The transformation from SDPT to silver HCA consists of three phases:

1. Use a mapping of dependency relations to clause constituents: subjects (S) and the governor, i.e., a non-copular verb(V), via relation `nsubj` and such; objects (O) and complements (C) in V's dependants via relations `dobj`, `iobj`, `xcomp`, `ccomp`, and such; adverbials (A) in V's dependents via relations `advmod`, `advcl`, `prep_in`, and such.
2. When detecting a verb[5] in the sentence, a corresponding clause, consisting of the verb and its dependant constituents, can be identified.

[5] Note that a copular verb in a clause and other constituents are dependants of the complement.

3. If a clause governs another clause via a dependency relation, the interrelation between them can be determined by the relation label:

   - Coordination: `conj:and`, `conj:or`, `conj:but`
   - Subjective: `nsubj`
   - Objective: `dobj`, `iobj`
   - Predicative: `xcomp`, `ccmop`
   - Appositive: `appos`
   - Relative: `ccomp`, `acl:relcl`, `rcmod`
   - Adverbial: `advcl`

As demonstrated in Figure 5, the first two clauses of the sentence exemplified in Section 1 are identified through their syntactic dependency tree. Moreover, the inter-clause relation can be inferred as *Adverbial: Conditional* with the dependency relation `advcl` and the subordinator "If".

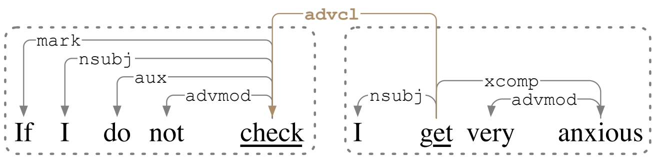


Figure 5. Extract clauses and inter-clause relations via the syntactic dependency parse tree. Two clauses in dashed boxes are identified by the underlined verb and the governed constituents. The inter-clause relation *Adverbial* can be determined by the dependency `advcl` between the two clauses.

3.3.2. Gold Data from Manual Annotator

As discussed above, the syntactic structures of CPT and SDPT can be transformed into clauses and inter-clause relations. However, these silver annotations are still unable to fulfill the need to build an HCA corpus for the following reasons:

- Specific inter-clause relations can not be obtained via the two syntactic structures, where CPT can only provide the existence of a relation without a label, and the dependency relations in SDPT have multiple mappings (e.g., `ccomp` to *Predcative* or *Relative*) or no mapping (e.g., `advcl` to no exact *Adverbial* subtype like *Conditional*).
- Pre-set transformation rules identify more clauses out of the HCA definitions. For example, the extracted non-finite clauses (e.g., *to*-infinitives) embedded in its matrix clause are too short and lead to hierarchical redundancies in the HCA tree.
- The performances of two syntactic parsers degrade when encountering long and complex sentences which are the core concerns of our HCA corpus.

Therefore, we recruit a group of human annotators with prior English grammar research experience to proofread these silver HCA on a browsed-based software `ClausAnn` created for the annotation work. The Java Web application `ClausAnn` provides convenient operations and efficient keyboard shortcuts for annotators, and we open source it on our GitHub repository[6]. A typical annotation trial on `ClausAnn` consists of the following steps:

1. Review annotations from CPT, SDPT, or other annotators by switching the name tags in Figure 6(a).
2. Choose an existing annotation to proofread or just start from the original sentence.
3. Segment a text span into two by double-clicking the blank space of a split point, and select the relation between them in Figure 6(b).

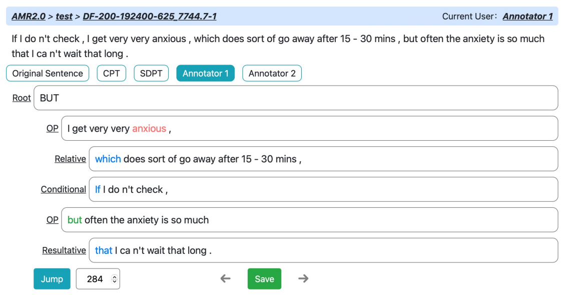
   (a) If the two spans are coordinated, select a specific coordination and label coordinators that indicate the interrelation in Figure 6(c).

---

[6] https://github.com/MetroVancloud/ClausAnn

(b) If the two spans are subordinated, designate the superordinate one, select a specific subordination, and label subordinators that indicate the interrelation in Figure 6(d).

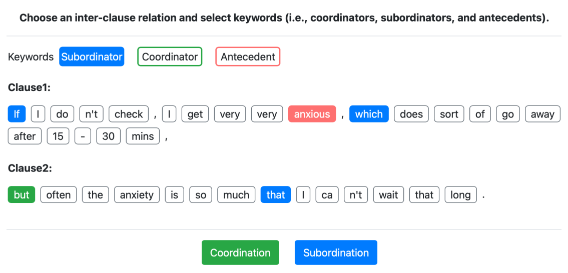
4. Remerge two text spans into one by clicking the two spans successively.
5. Repeat step 3 and 4 until all text spans are segmented into a set of clauses constructed in a right HCA tree.


AMR2.0 > test > DF-200-192400-625_7744.7-1 Current User: Annotator 1

If I do n't check , I get very very anxious , which does sort of go away after 15 - 30 mins , but often the anxiety is so much that I ca n't wait that long .

Root: BUT
OP: I get very very anxious ,
Relative: which does sort of go away after 15 - 30 mins ,
Conditional: If I do n't check ,
OP: but often the anxiety is so much
Resultative: that I ca n't wait that long .

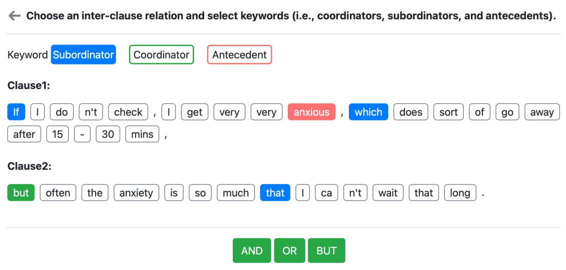
(a) Switch annotator labels and review the corresponding annotation


Choose an inter-clause relation and select keywords (i.e., coordinators, subordinators, and antecedents).

Clause1:
If I do n't check , I get very very anxious , which does sort of go away after 15 - 30 mins ,

Clause2:
but often the anxiety is so much that I ca n't wait that long .

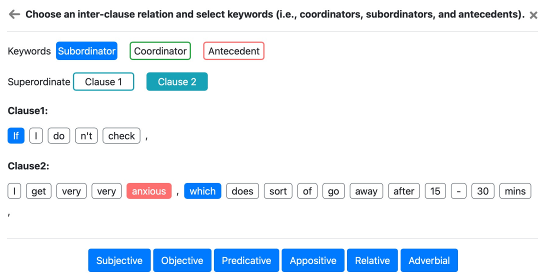
(b) Segment a text span into two, and choose a coordination or subordination between them.


Choose an inter-clause relation and select keywords (i.e., coordinators, subordinators, and antecedents).

Clause1:
If I do n't check , I get very very anxious , which does sort of go away after 15 - 30 mins ,

Clause2:
but often the anxiety is so much that I ca n't wait that long .

(c) When choosing coordination, select the exact coordinate relation, i.e., *AND*, *OR*, or *BUT*


Choose an inter-clause relation and select keywords (i.e., coordinators, subordinators, and antecedents).

Superordinate: Clause 1 Clause 2

Clause1:
If I do n't check ,

Clause2:
I get very very anxious , which does sort of go away after 15 - 30 mins ,

(d) When choosing subordination, select the superordinate clause and the exact subordinate relation, i.e., *Subjective*, *Objective*, and such

Figure 6. Operating steps of an annotation trial in the browser-based tool, `ClausAnn`

3.3.3. Quality Assurance

There are mainly two steps taken jointly to ensure the quality of the final HCA corpus, i.e., multi-round annotation and consistency measurement.

The total annotation work consists of 39, 260 sentences, and three rounds of annotation are arranged by 5%, 5%, and 90% of total sentences and conducted with a progressive and negotiable strategy. Before the first round, every annotator thoroughly understands the HCA framework and uses the tool `ClausAnn` proficiently after adequate hands-on training. During the first two rounds, the lead annotator, who majors in English grammar, organizes a discussion on complex or abnormal cases with other annotators.

For consistency measurement, we tracked inter-annotator agreement (IAA) after each round of the annotation work. As discussed in Section 2.1, the HCA and RST parsing tasks aim to extract the clause/EDU hierarchy from texts. Thus, the evaluation metrics of discourse segmentation [15] and discours parsing [16] subtasks in RST parsing are adopted as IAA metrics in evaluating the annotation quality of the HCA corpus:

- P/R/F_1 on clauses: precision, recall, and F_1 score on the segmented clauses, where a positive match means that both segmented clauses from two annotators have the same start and end boundaries.
- RST-Parseval [31] on interrelations: consisting of *Span*, *Nuclearity*, *Relation*, and *Full* are used to evaluate unlabeled, nuclearity-, relation-, and fully labeled interrelations respectively between the matched clauses from two annotators.

In the first two rounds of annotation, a total of 10% sentences are double-annotated, and the ratio comes to 16% in the last round, higher than 13.8% in the RST-DT corpus [7]. According to the statistics, the IAA measured by the above two metrics grows as the annotation rounds increase, indicating that the two steps of multi-round annotation and consistency measurement play a significant role in ensuring annotation quality.

As shown in Table 5, the final IAA achieves high consistencies, where

- F_1 scores on clauses range from 98.4 to 100,
- RST-Parseval scores on interrelations range from 97.3, 97.0, 93.6, 93.4 to 98.1, 97.8, 94.2, 94.1 in *Span*, *Nuclearity*, *Relation*, and *Full*, respectively.

Table 5. Inter-annotation agreement (IAA) of 16% double-annotated sentences in the HCA corpus by ten annotators marked as 1 to 10. Note that **bold** and underlined figures indict the highest and lowest consistencies in the corresponding metrics, respectively.

Annotator	**Clause**			**Interrelation**			
	P	**R**	**F_1**	**Span**	**Nuc.**	**Rel.**	**Full**
1, 2	99.9	99.8	99.8	97.9	97.5	**94.3**	**94.1**
1, 3	**100**	**100**	**100**	**98.1**	**97.8**	94.2	**94.1**
2, 4	99.8	99.5	99.6	97.5	97.3	93.9	93.8
1, 5	99.0	98.3	98.6	98.0	97.7	94.2	94.0
6, 7	99.6	99.1	99.3	97.3	97.0	93.8	93.6
4, 8	99.4	99.3	99.3	97.9	93.9	93.7	93.5
1, 9	99.0	98.6	98.8	97.2	97.0	93.9	93.8
5, 10	98.8	98.0	98.4	97.3	97.1	93.6	93.4

Compared with the RST-DT corpus, whose IAA score on EDUs ranges from 95.1 to 100 and IAA scores on rhetorical relations with three metrics, *Spans*, *Nuclearity*, *Relation*, range from 77.8, 69.5, 59.7 to 92.9, 88.2, 79.2, our HCA corpus reaches better consistencies, as the HCA framework has more restrict definitions on the elementary unit (i.e., clauses) and fewer types of the interrelation.

3.3.4. Dataset Detail

The resulting HCA-AMR2.0 dataset is based on AMR 2.0, which contains 39,260 sentences, and 19,376 (49.4%) sentences are paired with an HCA tree, while the rest are simple sentences with only one clause. The train, dev, and test set split follows the original split in AMR 2.0. Detailed statistics are listed in Table 6.

Table 6. Main statistics of the Hierarchical Clause Annotation dataset based on AMR 2.0 (HCA-AMR2.0). * means that some input sequences contain multiple sentences, and the coordination *MulSnt* is necessary for these inter-sentence relations in these cases. * indicates that *Adverbial* can be divided into nine sub-types like *Condition*, *Concession*, and *Purpose*.

Item	**Occurence**	**Relation**	**Occurence**
# Sentences (S)	39,260	*MulSnt**	3,249
# S with HCA	19,376	*And*	14,952
# S in Train Set	17,885	*Or*	974
# S in Dev Set	740	*But*	3,704
# S in Test Set	751	*Subjective*	992
# Tokens (T)	521,805	*Objective*	8,741
# Clauses (C)	*57,330*	*Predicative*	1,009
# Avg. T/S	26.9	*Appositive*	667
# Avg. T/C	9.1	*Relative*	7,095
# Avg. C/S	3.1	*Adverbial**	7,777

4. Model

In this paper, we model hierarchical clause annotation (HCA) as a two-stage task, i.e., clause segmentation and clause parsing, and provide auto-annotation baselines for each subtask. Clause segmentation segments a complex sentence into several clauses, while clause parsing links the clauses with interrelations into a clause tree.

4.1. Clause Segmentation

We encounter clause segmentation task with a sequence tagging model, DiscoDisCo [15] used in the discourse segmentation task: embed the input sentence, encode with a single Bi-LSTM, decode with a linear projection layer, and indicate the first token of each clause in the output tag sequence.

In the embedding layer, we rely on three kinds of word embeddings concatenated together:

(1) Bi-LSTM encoded character embeddings,
(2) static word embeddings from fastText, [32]
(3) and fine-tuning word embeddings from a pretrained language model (PLM).

Additionally, we introduce and embed various grammatical information like lemmas, part-of-speech (POS), and syntactic dependencies generated by Stanza [30]. These tokenwise features embeddings are concatenated with word embeddings, and the word w_i in the input sentence is embedded as

$$u_i = \text{Concat}(u_i^{word} + u_i^{feat}) \tag{1}$$

where u_i^{word} is concatenated from three kinds of word embeddings and u_i^{feat} is grammatical features embeddings.

The embedding $\mathbf{u}_{1:n}$ of the input sentence with n tokens are fed through a BiLSTM network, and the output s_i is then calculated by a linear projection layer to predict the segmentation tag for i-th token:

$$s_i = \text{Linear}(\text{BiLSTM}(\mathbf{u}_{1:n}, i)). \tag{2}$$

Given s_i, we get the predicted tag sequence $\hat{\mathbf{t}}$, and optimizing the cross-entropy loss of $\mathcal{L}(\hat{\mathbf{t}}, \mathbf{t})$ to train the weights of the PLM, the BiLSTM network and the linear projection layer, where $\mathbf{t}$ is the gold tag sequence.

4.2. Clause Parsing

Given segmented clauses in the input sentence, we model clause parsing as a discourse parsing task, where clauses are considered as EDUs and linked into a clause tree. Therefore, we employ discourse parsers in [16], which contain a span-based parser with the top-down strategy and a shift-reduce transition parser with the bottom-up strategy for their simple architectures and open codes. Overviews of the parsers are shown in Figures 7 and 8. Note that the two parsing strategies share the same word embedding layer to represent text spans.

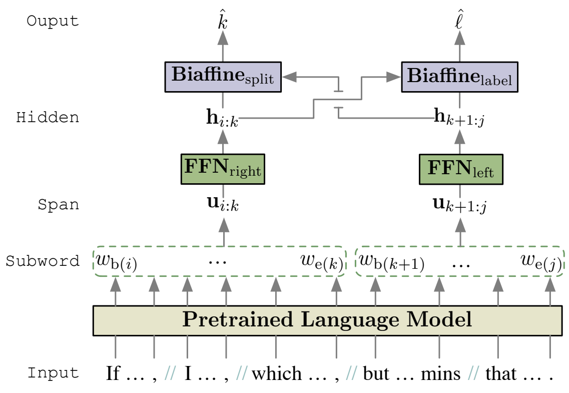


Figure 7. Top-Down Clause Parsing.

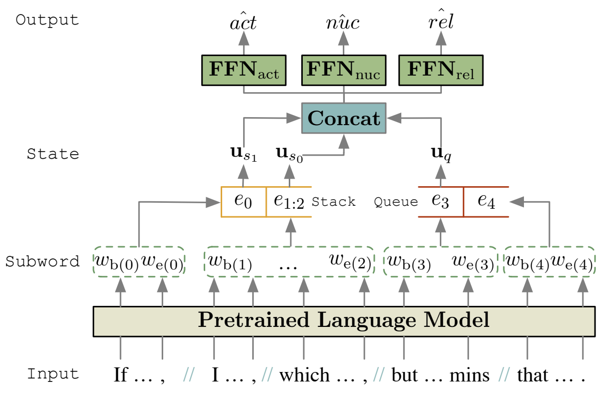


Figure 8. Bottom-Up Clause Parsing.

4.2.1. Text Span Embedding

In the process of clause parsing, the representation of text spans is needed for either "span-to-clause" splitting in the top-down parsing strategy or "clause-to-span" combining in the bottom-up parsing strategy. Therefore, we transform the input sentence into a subword sequence $\{t_1, t_2, \dots t_n\}$, and obtain the embedding $\{w_1, w_2, \dots w_n\}$ using a PLM. The embedding for a text span $\mathbf{u}_{i:j}$, consisting of the i-th clause to the j-th clause, is obtained by averaging the vector of both edge subwords,

$$\mathbf{u}_{i:j} = (w_{\text{b}(i)} + w_{\text{e}(j)})/2, \tag{3}$$

where $\text{b}(i)$ returns the index of the **b**egin subword in the i-th clause and $\text{e}(j)$ returns that of the **e**nd subword in the j-th clause.

4.2.2. Top-Down Strategy

The top-down parser splits each span into smaller ones recursively until the span becomes a single clause. We introduce biaffine networks [33] for span splitting and a loss penalty.

For each position k in a span consisting of the i-th clause to the j-th clause, a scoring function $s_{\text{split}}(i,j,k)$, is defined as follows:

$$s_{\text{split}}(i,j,k) = \mathbf{h}_{i:k}\mathbf{W}\mathbf{h}_{k+1:j} + \mathbf{v}_{\text{left}}\mathbf{h}_{i:k} + \mathbf{v}_{\text{right}}\mathbf{h}_{k+1:j} \tag{4}$$

where $\mathbf{W}$, $\mathbf{v}_{\text{left}}$ and $\mathbf{v}_{\text{right}}$ are weight matrices in the biaffine layer for splitting a text span. Here, $\mathbf{h}_{i:k}$ and $\mathbf{h}_{k+1:j}$ are defined as follows:

$$\mathbf{h}_{i:k} = \text{FFN}_{\text{left}}(\mathbf{u}_{i:k}), \tag{5}$$

$$\mathbf{h}_{k+1:j} = \text{FFN}_{\text{right}}(\mathbf{u}_{k+1:j}) \tag{6}$$

Then, the span is split at the end of an inner clause that maximizes Equation 4:

$$\hat{k} = \arg\max_{i \le k < j} s_{\text{split}}(i,j,k). \tag{7}$$

When splitting a span at the end of the $\hat{k}$-th clause, the score of the nuclearity and relation labels for the two spans is defined as follows:

$$s_{\text{label}}(i,j,\hat{k},\ell) = \mathbf{h}_{i:\hat{k}}\mathbf{W}^{\ell}\mathbf{h}_{\hat{k}+1:j} + \mathbf{v}^{\ell}_{\text{left}}\mathbf{h}_{i:\hat{k}} + \mathbf{v}^{\ell}_{\text{right}}\mathbf{h}_{\hat{k}+1:j} \tag{8}$$

where $\mathbf{W}^{\ell}$, $\mathbf{v}^{\ell}_{\text{left}}$, and $\mathbf{v}^{\ell}_{\text{right}}$ are weight matrices in the biaffine layer for predicting an inter-clause relation. Then, the label that maximizes Equation 8 is assigned to the spans:

$$\hat{\ell} = \arg\max_{\ell \in \mathcal{L}} s_{\text{label}}(i, j, \hat{k}, \ell) \tag{9}$$

where $\mathcal{L}$ denotes three nuclearity labels, $\{\xrightarrow{sub.}, \xleftarrow{sub.}, \xleftrightarrow{co.}\}$, for predicting the nuclearity, and a set of inter-clause relations labels for predicting the exact relation. Note that the parameters in biaffine layers and FFNs for the nuclearity and relation labeling are learned separately.

4.2.3. Bottom-Up Strategy

Formally, in a shift-reduce model with a bottom-up strategy, the parsing state is denoted as a tuple(S, Q), where S is a stack that stores processed clauses and Q is a queue that contains incoming clauses. Each element in S can be an unreduced clause e_i or a combined composite item $e_{i:j}$. At each step, the parser chooses one of the following actions with an FFN classifier and updates the state (S, Q):

- $SHIFT$: pop the first clause off Q and push it onto S.
- $REDUCE$: pop two elements from S and push a new combined composite item that has the popped subtrees as its children onto S as a single composite item.

We employ three FFN classifiers, FFN_{act}, FFN_{nuc}, and FFN_{rel}, where FFN_{act} predicts an action, and the rest two decide the nuclearity and the label of an inter-clause relation after a $REDUCE$ action. Specifically, the output dimension of FFN_{act} is 2 ($SHIFT$ or $REDUCE$), that of FFN_{nuc} is 3 ($\xrightarrow{sub.}$, $\xleftarrow{sub.}$ or $\xleftrightarrow{co.}$), and that of FFN_{rel} is the number of inter-clause relations defined in the HCA framework. Three classifier outputs s_* are defined as

$$s_* = \text{FFN}_*(\text{Concat}(\mathbf{u}_{s_0}, \mathbf{u}_{s_1}, \mathbf{u}_{q_0})) \tag{10}$$

where function Concat concatenates three state vectors: $\mathbf{u}_{s_0}$ is the representation of the top clause stored in S, $\mathbf{u}_{s_1}$ is that in the second clause of S, and $\mathbf{u}_{q_0}$ is that in the first clause in Q. Weights of the PLM and each FFN are trained by optimizing the cross-entropy loss of s_{act}, s_{nuc}, and s_{rel}.

5. Experiments

In this section, we elaborate on the experimental details of the proposed baseline models for two HCA subtasks, clause segmentation, and clause parsing, on the novel HCA-AMR2.0 corpus.

5.1. Dataset

Except for the HCA-AMR2.0 corpus, we reference three discourse analysis corpora, GUM [34], STAC [35], and RST-DT [7], to further verify the effectiveness of the proposed models and expound the distinctive advantages of HCA-AMR2.0 over these discourse analysis corpora.

- *HCA-AMR2.0:* The first HCA corpus is annotated on sentences in AMR 2.0 dataset. The source data includes discussion forums collected for the DARPA BOLT AND DEFT programs, transcripts and English translations of Mandarin Chinese broadcast news programming from China Central TV, Wall Street Journal text, translated Xinhua news texts, various newswire data from NIST OpenMT evaluations and weblog data used in the DARPA GALE program.
- *GUM:* The Georgetown University Multilayer corpus is created as part of the course LING-367 (Computational Corpus Linguistics) at Georgetown University, and its annotation is followed the RST-DT segmentation guidelines for English. The text sources consist of 35 documents of news, interviews, instruction books, and travel guides from WikiNews, WikiHow, and WikiVoyage.

- *STAC:* The Strategic Conversation dataset is a corpus of strategic chat conversations in 45 games annotated with negotiation-related information, dialogue acts, and discourse structures in Segmented Discourse Representation Theory (SDRT) framework.
- *RST-DT:* Rhetorical Structure Theory (RST) Discourse Treebank is developed by researchers at the Information Sciences Institute (University of Southern California), the US Department of Defense, and the Linguistic Data Consortium (LDC). It comprises 385 Wall Street Journal articles from the Penn Treebank annotated with discourse structure in the RST framework.

For the clause segmentation subtask, we select all three discourse analysis datasets evaluated on the discourse segmentation task as a comparison, and the data files are all in `conll` formats. For the clause parsing subtask, we only select the RST-DT dataset evaluated on the discourse parsing task as a comparison, as previous works of discourse parsing experiments on the other two datasets using *accuracy* as the evaluating metric other than *Parseval*. The interrelation data files in HCA and RST-DT are preprocessed by Heilman and Sagae's system [36]. The important dataset statistics related to the experiments are listed in Table 7.

Table 7. Main statistics of HCA-AMR2.0, GUM, STAC, and RST-DT dataset. Note that *Unit* in the header represents clause or elementary discourse unit (EDU), *Rel.* represents inter-clause/EDU relation, *#Avg. U/S* means the average number of units per sentence, and *#Avg. R/U* means the average number of inter-clause/EDU relations per unit.

Dataset	**#Unit (U) / #Sentence (S)**			**#Avg. U/S**	**#Rel.**	**#Rel. Type**	**#Avg. Rel./U**
	Train	**Dev**	**Test**				
HCA-AMR2.0	52,758/17,885	2,222/740	2,350/751	3.1	49,160	18	0.86
GUM	14,766/6,346	2,219/976	2,283/999	2.3	-	-	
STAC	9,887/8,754	1,154/991	1,547/1,342	1.1	-	-	-
RST-DT	17,646/6,671	1,797/716	2,346/928	2.6	19,778	18	0.91

5.2. Experimental Environments

The information on the main hardware and software used in our experimental environments is listed in Table 8.

Table 8. Hardware and software used in our experiments.

Environment	**Value**
Hardware	
CPU	Intel i9-10900K @ 3.7 GHz (10-core)
GPU	NVIDIA RTX 3090Ti (24G)
Memory	64 GB
Software	
Python	3.8.16
Pytorch	1.12.1
Anaconda	4.10.1
CUDA	11.3
IDE	PyCharm 2022.2.3

5.3. Hyper-Parameters

For the hyper-parameters in the models for clause segmentation and clause parsing subtasks, we list their layer, name, and value in Table 9. Note that both models are not trained until reaching the maximum epochs, where the model of clause segmentation is trained in about thirteen epochs for five hours, and the model of clause parsing is finished in about seven epochs for ten hours with the experimental environments introduced in Section 5.2.

Table 9. Final hyper-parameters configuration of clause segmentation model.

Layer	Hyper-Parameter	Value
Clause Segmentation Model		
Character Embedding (Bi-LSTM)	layer	1
	hidden_size	64
	dropout	0.2
Word Embedding	fastText	300
	Electra	1024 (large)
Feature Embedding	POS/Lemma/DP	100
Bi-LSTM	layer	1
	hidden_size	512
	dropout	0.1
Trainer	optimizer	AdamW
	learning rate	5e-4, 1e-4
	# epochs	60
	patience of early stopping	10
	validation criteria	+span_f1
Clause Segmentation Model		
Word Embbeding	pretrained language model	768/1024(base/large)
FFN	hidden_size	512
	dropout	0.2
Trainer	optimizer	AdamW
	learning rate	2e-4, 1e-5
	weight decay	0.01
	batch size (# spans/actions)	5
	# epochs	20
	patience of early stopping	5
	gradient clipping	1.0
	validation criteria	RST-Parseval-*Full*

5.4. Evaluation Metrics

As described in Section 3.3.3, we introduce two metrics for IAA to evaluate annotation consistencies between two annotators. Thus, we still use $P/R/F_1$ on clauses for clause segmentation subtask and *RST-Parseval* on inter-clause relations for clause parsing subtask to evaluate the consistencies between gold and predicted annotations.

5.5. Baseline Models

We aim to utilize models proposed by previous works on discourse segmentation and discourse parsing tasks and experiment with our HCA-AMR2.0 corpus to obtain effective baseline models for the novel clause segmentation and parsing subtasks.

5.6. Experimental Results

The experiments of clause segmentation and parsing subtasks are conducted separately, and experimental results are compared with those of previous works on discourse segmentation and parsing tasks, respectively.

5.6.1. Results of Clause Segmentation

We adapt the discourse parser DisCoDisCo [15] into the clause segmentation task and conduct an ablation study on different features, e.g., lemma, syntactic dependency, part-of-speech, and static word embedding fastText, to explore which contributes more to the performances. The experimental results of the clause segmentation task are reported in Table 10, as well as the performances of previous works on the discourse segmentation task for comparison. From the results, we have the following observations:

- Compared with the 98.4-100 IAA scores on clause segmentation mentioned in Section 3.3.3, the adapted DisCoDisCo model achieves satisfactory performances, i.e., 91.3 F_1 scores.
- In the ablation study of different embedded features, the static word embedding fastText, which gains a 4.9 F_1 score improvement, contributes the most in all features, although other features also have positive impacts on the performances.
- In the discourse segmentation task, the DisCoDisCo model outperforms the GumDrop model in the GUM and RST-DT datasets. Although GumDrop performs better than DisCoDisCo in the STAC dataset, Its performance sharply declined with 14.7 F_1 scores when removing extra gold features. Thus, we choose to apply the DisCoDisCo model to the clause segmentation subtask.
- Experimented with the same model DisCoDisCo, performances on clause segmentation are about 3-5 F_1 scores lower than on discourse segmentation, indicating that the performance gaps are attributed to the corpora. From the statistics in Table 6, an average of 3.1 clauses constitute a sentence in HCA-AMR2.0, more than that in GUM (2.3 EDUs), STAC (1.1 EDUs), and RST-DT (2.6 EDUs). Besides, only HCA-AMR2.0 contains a certain number of weblog data, which are less formal in English grammar and may cause more obstacles for the DisCoDisCo model.

Table 10. Performaces of the adapted DisCoDisCo model on HCA-AMR2.0 for clause segmentation, and performances of DisCoDisCo and GumDrop on three datasets for the contrastive task, discourse segmentation. Note that * and ° indicate gold annotated features from the corresponding dataset and silver features annotated by Stanza, respectively. **Bold** numbers are the best scores in each dataset. All the experiments on the clause segmentation task are conducted for five runs with different seeds, and the experimental results are averaged.

Task	Dataset	Model	P	R	F_1
Discourse Segmentation	GUM	GumDrop[37]	96.5	90.8	93.5
		*- all feats.**	**97.7**	87.4	92.3
		DisCoDisCo[15]	93.9	94.4	**94.2**
		*- all feats.**	92.7	**92.6**	92.6
	STAC	GumDrop[37]	95.3	**95.4**	**95.3**
		*- all feats.**	85.0	76.7	80.6
		DisCoDisCo[15]	**96.3**	93.6	94.9
		*- all feats.**	91.8	92.1	91.9
	RST-DT	GumDrop[37]	94.9	96.5	95.7
		*- all feats.**	96.3	94.6	95.4
		DisCoDisCo[15]	96.4	**96.9**	**96.6**
		*- all feats.**	**96.8**	95.9	96.4
Clause Segmentation	HCA-AMR2.0	DisCoDisCo	**92.9**	89.7	**91.3**
		- lem.°	86.8	**93.9**	90.2
		- dp.°	91.0	87.7	89.3
		- pos°	91.4	87.6	89.4
		- all feats.°	89.2	85.4	87.2
		- fastText	90.5	82.7	86.4

5.6.2. Results of Clause Parsing

We employ the bottom-up and top-down discourse parsers in [16] as our clause parsing models and conduct experimental trials of the base version of different pretrained language models (PLMs, i.e., BERT [38], RoBERTa [39], SpanBERT [40], XLNet [41], and DeBERTa [42]) to obtain better performances. Table 11 demonstrates the main results of both bottom-up and top-down models on clause parsing, as well as that on discourse parsing for comparison. From the results, we have the following observations:

- Better performances are obtained in clause parsing than in discourse parsing by either top-down or bottom-up parsers with whatever PLMs.
- The best performances of clause parsing are obtained by the bottom-up parser with the pretrained DeBERTa, where the performance reaches up to 97.0 F_1 scores on *Parseval-Span* and falls to 87.7 F_1 scores on *Parseval-Full*.
- All the best performances in either clause parsing or discourse parsing are obtained by parsers with pretrained XLNet and DeBERTa, indicating that these two PLMs are more suitable for relation classification tasks than other PLMs.
- Experimented with a same model and a same pretrained language model, performances on clause parsing are about 6-9 *Parseval-Full* scores higher than that on discourse parsing, indicating that the performance gaps are attributed to the differences between corpora. RST-DT contains 18 classes of relations partitioned from 78 types of blurry rhetorical relations, while HCA has 18 types of distinguishable semantic relations, half of which are subdivided from *Adverbial*.

Table 11. Performances of the top-down and bottom-up parsers with various pretrained language models (PLMs) for clause parsing and discourse parsing tasks, which are evaluated by four RST-Parseval metrics, i.e., *Span*, *Nuclearity* (Nuc.), *Relation* (Rel.), and *Full*. Standard deviations for three runs are shown in parentheses. **Bold** numbers are the best scores in each task with each model. Note that we only conduct experiments on HCA-AMR2.0 and RST-DT datasets for clause parsing and discourse parsing, respectively.

Model	**PLM**	**Discourse Parsing**				**Clause Parsing**			
		Span	**Nuc.**	**Rel.**	**Full**	**Span**	**Nuc.**	**Rel.**	**Full**
Top-Down	BERT	92.6±0.53	85.7±0.41	75.4±0.45	74.7±0.54	96.0±0.20	92.1±0.27	85.8±0.51	85.7±0.47
	RoBERTa	94.1±0.46	88.4±0.46	79.6±0.17	78.7±0.11	96.5±0.02	93.1±0.09	87.1±0.23	87.1±0.22
	SpanBERT	94.1±0.15	88.8±0.19	79.4±0.49	78.5±0.39	96.4±0.13	92.6±0.12	86.1±0.15	85.9±0.15
	XLNet	**94.8±0.39**	**89.5±0.39**	**80.5±0.59**	**79.5±0.53**	96.6±0.07	**93.5±0.20**	87.0±0.57	86.9±0.56
	DeBERTa	94.2±0.33	89.0±0.16	80.1±0.43	79.1±0.32	**96.6±0.02**	93.3±0.07	**87.2±0.04**	**87.2±0.10**
Bottom-Up	BERT	91.9±0.34	84.4±0.31	74.4±0.37	73.8±0.30	96.3±0.22	92.3±0.36	86.1±0.37	86.0±0.32
	RoBERTa	94.4±0.12	89.0±0.34	80.4±0.47	79.7±0.51	96.4±0.16	92.8±0.20	86.6±0.56	86.6±0.58
	SpanBERT	93.9±0.24	88.2±0.19	79.3±0.37	78.4±0.29	96.5±0.09	92.6±0.06	86.4±0.20	86.2±0.22
	XLNet	**94.7±0.31**	89.4±0.24	**81.2±0.27**	**80.4±0.34**	96.9±0.10	93.6±0.09	87.4±0.33	87.3±0.31
	DeBERTa	94.6±0.38	**89.8±0.65**	81.0±0.64	80.2±0.70	**97.0±0.10**	**94.0±0.17**	**87.8±0.39**	**87.7±0.38**

To obtain a clause parser with better performances, we conduct experimental trials with the large versions of PLMs, and the results are illustrated in Table 12. As can be observed from the results,

- All the parsers with the corresponding large PLMs perform better than that with the base PLMs.
- The bottom-up parser with DeBERTa-large achieves a better performance, with a 0.8 F_1 score improvement on *Parseval-Full* over the parser with DeBERTa-base.

Table 12. Clause Parsing Results with large versions of pretrained language models (PLMs), XLNet, and DeBERTa (RST-Parseval). † indicates PLMs with a large version. Standard deviations for three runs are shown in parentheses. **Bold** numbers are the best scores in each model.

Model	PLM	Span	Nuc.	Rel.	Full
Top-Down	XLNet	96.6±0.07	93.5±0.20	87.0±0.57	86.9±0.56
	XLNet†	96.7±0.27	93.6±0.33	**87.6±0.36**	**87.6±0.37**
	DeBERTa	96.6±0.02	93.3±0.07	87.2±0.04	87.2±0.10
	DeBERTa†	**97.0±0.14**	**94.0±0.29**	87.6±0.69	87.6±0.61
Bottom-Up	XLNet	96.9±0.10	93.6±0.09	87.4±0.33	87.3±0.31
	XLNet†	97.0±0.34	93.7±0.46	87.6±0.67	87.6±0.67
	DeBERTa	97.0±0.10	94.0±0.17	87.8±0.39	87.7±0.38
	DeBERTa†	**97.4±0.08**	**94.5±0.13**	**88.6±0.27**	**88.5±0.28**

6. Discussion

6.1. Potentialities of HCA

As discussed in Section 1, we demonstrated the same hierarchy of the HCA tree, the AMR graph, and the SDG of a complex sentence, indicating the potentialities of utilizing the structural information of the HCA tree to improve semantic parsing. Thus, we provide two case studies where simple transformation rules derived from the HCA tree can be applied to these two semantic parsing tasks.

6.1.1. Case Study for AMR Parsing

For AMR parsing, we employ the state-of-the-art AMR parser proposed in [1] to predict the AMR graph of the exemplified sentence in Section 1. As shown in Figure 9, two dotted red edges are missed by the parser when compared with the gold AMR, while two solid red edges are mistakenly predicted.

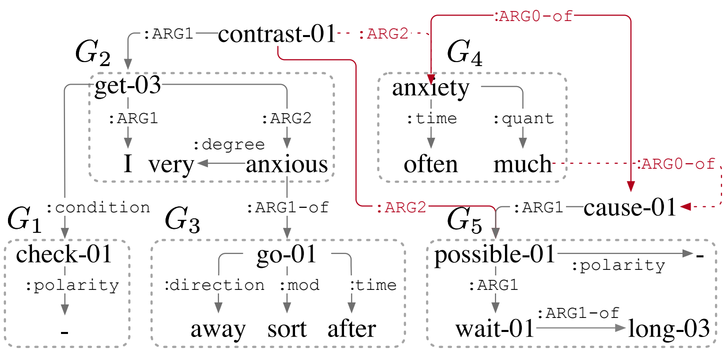


Figure 9. Abstract Meaning Representation (AMR) Graph predicted by the state-of-the-art AMR parser. Red dotted relation edges, which are missed by the parser, can be recovered by transformation rules derived from the HCA tree. Red solid relation edges, which are mistakenly predicted by the parser, can be deleted by transformation rules derived from the HCA tree.

From the HCA tree given in Figure 1(a), the clause *"I get very anxious "* mapping to the subgraph G_2 and the clause *"but often the anxiety is so much"* mapping to subgraph G_4 are coordinate and contrastive. Therefore, we provide a transformation rule:

1. Transform the inter-clause relation `BUT` to an AMR node `contrast-01` and two AMR edges directing to the root nodes of G_2 and G_4.

Meanwhile, the clause *"that I can not wait that long"* mapping to subgraph G_5 is a resultative adverbial clause subordinated to the clause *"but often the anxiety is so much"* mapping to the subgraph G_4. Note that the verb of the matrix clause is copular *"is"*, and the subordinate clause modifies the complement *"much"* in the matrix clause. Therefore, a new transformation rule can be derived:

2. Transform the inter-clause relation `resultative` to an AMR node `cause-01` and two AMR edges directing to the root node of G_5 and the node `much` in G_4.

With these two transformation rules, we can delete the solid red edges which are mistakenly predicted and add the dotted red edges which are missed.

6.1.2. Case Study for Semantic Dependency Parsing

For semantic dependency parsing, we employ the state-of-the-art AMR parser proposed in [2] to predict the semantic dependency graph (SDG) of the exemplified sentence in Section 1. As shown in Figure 10, three dotted red edges are missed by the parser when compared with the gold SDG.

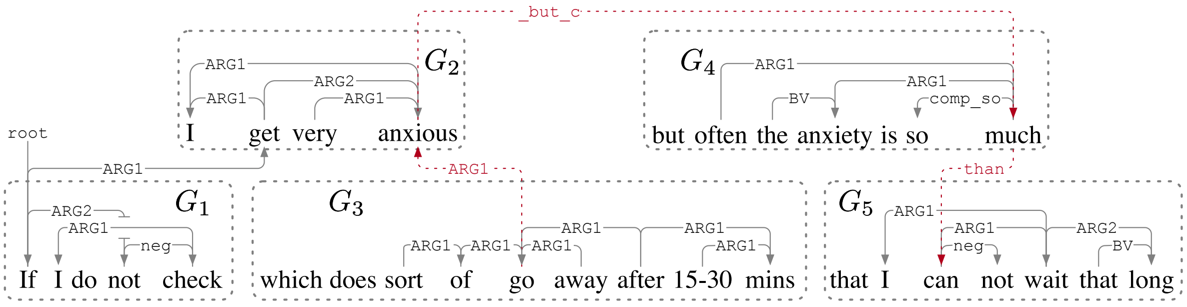


Figure 10. Semantic dependency graph (SDG) predicted by the state-of-the-art semantic dependency parser, `DynGL-SDP`. Dotted red dependency edges, which are missed by the parser, can be recovered by transformation rules derived from the HCA tree.

As dicussed in Section 6.1.1, inter-clause relations among three clauses mapping to subgraph $G2$, $G4$, and $G5$ can be derived into the following transformation rules:

1. Transform the inter-clause relation `BUT` to a dependency edge between root nodes (i.e., `anxious` and `much`) of G_2 and G_4.
2. Transform the inter-clause relation `resultative` to a dependency edge between root nodes (i.e., `much` and `can`) of G_4 and G_5.

Moreover, the relative subordinate clause "*which does sort of go away after 15-30 mins*" mapping to subgraph G_3 modifies the complement "anxious" in the matrix clause "*I get very anxious*" mapping to subgraph G_2. Thus, a new transformation rule can be derived:

3. Transform the inter-clause relation `relative` to a dependency edge between the root node `go` G_3 and the node `anxious`) of G_2.

With these three transformation rules, we can add the dotted red edges which are missed by the parser.

To sum up, we provide two case studies demonstrating the potentialities of HCA in semantic parsing, where transformation rules derived from the HCA tree are applied to modify two state-of-the-art parsers of the AMR parsing and semantic dependency parsing tasks.

6.2. Future Work

As demonstrated in the experimental results, we adapt discourse segmenter and discourse parser into our HCA subtasks (i.e., clause segmentation and parsing) and achieve satisfactory performances. However, there is still much room for improvement compared with IAA scores of manual annotation. Therefore, we aim to design better models for the HCA task in further research.

In the case studies, we derive some transformation rules from the HCA tree to modify the AMR graph and the SDG, which is explainable to humans but impractical. For this limitation, we aim to explore better ways of integrating the HCA structure in semantic parsing and more downstream NLP tasks.

7. Conclusions

In this paper, we propose a novel framework, hierarchical clause annotation (HCA), to segment complex sentences into clauses and capture inter-clause relations with strict definitions from linguistic research of clause hierarchy. We aim to explore the potentialities of integrating HCA structural features in semantic parsing with complex sentences, avoiding the deficiencies of previous works such as RST-parsing, SRRP, TS, SSD, etc. Following the HCA framework, we build up a large HCA corpus comprising $19,376$ English sentences from AMR 2.0. The annotation consists of silver data transformed from the constituency and syntactic dependency parse trees, and gold data annotated by experienced human annotators using a newly-created tool, `ClausAnn`. Moreover, we decompose HCA into two subtasks, i.e., clause segmentation and clause parsing, and provide effective baseline models for both subtasks to generate more HCA data.

Author Contributions: Conceptualization, Y. F. and Z. G.; Methodology, Y. F. and B. L.; Software, Y. F.; Annotation, all authors; Writing—original draft, Y. F.; Writing—review and editing, B. L., Y. S., M. G., C. S., S. C., and Z. G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used in our experiments are publicly available: Hierarchical Clause Annotation-AMR2.0 (HCA-AMR2.0) at https://github.com/MetroVancloud/HCASeg (accessed on 5 November 2022) and https://github.com/MetroVancloud/HCAParser (accessed on 5 November 2022), Abstract Meaning Representation (AMR) Annotation Release 2.0 (AMR 2.0) at https://catalog.ldc.upenn.edu/LDC2017T10 (accessed on 15 June 2021), RST Discourse Treebank (RST-DT) at https://catalog.ldc.upenn.edu/LDC2002T07 (accessed on 11 October 2022), The Georgetown University Multilayer corpus (GUM) at https://gucorpling.org/gum (accessed on 15 April 2023), Strategic Conversation (STAC) at https://www.irit.fr/STAC/ (accessed on 5 April 2023).

Acknowledgments: Our code extends two GitHub repositories https://github.com/gucorpling/DisCoDisCo (accessed on 1 March 2023) and https://github.com/nttcslab-nlp/RSTParser_EMNLP22 (accessed on 10 March 2023), thanks to them very much.

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper

Abbreviations

The following abbreviations are used in this manuscript:

AMR	Abstract Meaning Representation
BiLSTM	Bidirectional Long Short-Term Memory
CPT	Constituency Parse Tree
FFN	Feed Forward Neural Networks
HCA	Hierarchical Clause Annotation
IAA	Inter-Annotator Agreement
PLM	Pretrained Language Model
POS	Part-of-Speech
RST	Rhetorical Structure Theory
RST-DT	Rhetorical Structure Theory Discourse Treebank
SDG	Semantic Dependency Graph
SDPT	Syntactic Dependency Parse Tree
SPRP	Split-and-Rephrase
SSD	Simple Setence Decomposition
TS	Text Simplification

References

1. Sataer, Y.; Shi, C.; Gao, M.; Fan, Y.; Li, B.; Gao, Z. Integrating Syntactic and Semantic Knowledge in AMR Parsing with Heterogeneous Graph Attention Network. In Proceedings of the ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2023, pp. 1–5. https://doi.org/10.1109/ICASSP49357.2023.10097098.
2. Li, B.; Gao, M.; Fan, Y.; Sataer, Y.; Gao, Z.; Gui, Y. DynGL-SDP: Dynamic Graph Learning for Semantic Dependency Parsing. In Proceedings of the Proceedings of the 29th International Conference on Computational Linguistics; International Committee on Computational Linguistics: Gyeongju, Republic of Korea, 2022; pp. 3994–4004.
3. Tian, Y.; Song, Y.; Xia, F.; Zhang, T. Improving Constituency Parsing with Span Attention. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020; Association for Computational Linguistics: Online, 2020; pp. 1691–1703. https://doi.org/10.18653/v1/2020.findings-emnlp.153.
4. He, L.; Lee, K.; Lewis, M.; Zettlemoyer, L. Deep Semantic Role Labeling: What Works and What's Next. In Proceedings of the Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); Association for Computational Linguistics: Vancouver, Canada, 2017; pp. 473–483. https://doi.org/10.18653/v1/P17-1044.
5. Tang, G.; Müller, M.; Rios, A.; Sennrich, R. Why Self-Attention? A Targeted Evaluation of Neural Machine Translation Architectures. In Proceedings of the Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing; Association for Computational Linguistics: Brussels, Belgium, 2018; pp. 4263–4272. https://doi.org/10.18653/v1/D18-1458.
6. Xu, J.; Gan, Z.; Cheng, Y.; Liu, J. Discourse-Aware Neural Extractive Text Summarization. In Proceedings of the Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics; Association for Computational Linguistics: Online, 2020; pp. 5021–5031. https://doi.org/10.18653/v1/2020.acl-main.451.
7. Carlson, L.; Marcu, D.; Okurovsky, M.E. Building a Discourse-Tagged Corpus in the Framework of Rhetorical Structure Theory. In Proceedings of the Proceedings of the Second SIGdial Workshop on Discourse and Dialogue, 2001.
8. Narayan, S.; Gardent, C.; Cohen, S.B.; Shimorina, A. Split and Rephrase. In Proceedings of the Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing; Association for Computational Linguistics: Copenhagen, Denmark, 2017; pp. 606–616. https://doi.org/10.18653/v1/D17-1064.
9. Zhang, X.; Lapata, M. Sentence Simplification with Deep Reinforcement Learning. In Proceedings of the Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing; Association for Computational Linguistics: Copenhagen, Denmark, 2017; pp. 584–594. https://doi.org/10.18653/v1/D17-1062.
10. Gao, Y.; Huang, T.H.; Passonneau, R.J. ABCD: A Graph Framework to Convert Complex Sentences to a Covering Set of Simple Sentences. In Proceedings of the Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers); Association for Computational Linguistics: Online, 2021; pp. 3919–3931. https://doi.org/10.18653/v1/2021.acl-long.303.
11. Banarescu, L.; Bonial, C.; Cai, S.; Georgescu, M.; Griffitt, K.; Hermjakob, U.; Knight, K.; Koehn, P.; Palmer, M.; Schneider, N. Abstract Meaning Representation for Sembanking. In Proceedings of the Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse; Association for Computational Linguistics: Sofia, Bulgaria, 2013; pp. 178–186.
12. Oepen, S.; Kuhlmann, M.; Miyao, Y.; Zeman, D.; Cinková, S.; Flickinger, D.; Hajič, J.; Urešová, Z. SemEval 2015 Task 18: Broad-Coverage Semantic Dependency Parsing. In Proceedings of the Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015); Association for Computational Linguistics: Denver, Colorado, 2015; pp. 915–926. https://doi.org/10.18653/v1/S15-2153.
13. Mann, W.C.; Thompson, S.A. Rhetorical structure theory: Toward a functional theory of text organization. *Text-interdisciplinary Journal for the Study of Discourse* **1988**, *8*, 243–281.
14. Payne, T.E. *Understanding English Grammar: A Linguistic Introduction*; Cambridge University Press, 2010. https://doi.org/https://doi.org/10.1017/CBO9780511778988.

15. Gessler, L.; Behzad, S.; Liu, Y.J.; Peng, S.; Zhu, Y.; Zeldes, A. DisCoDisCo at the DISRPT2021 Shared Task: A System for Discourse Segmentation, Classification, and Connective Detection. In Proceedings of the Proceedings of the 2nd Shared Task on Discourse Relation Parsing and Treebanking (DISRPT 2021); Association for Computational Linguistics: Punta Cana, Dominican Republic, 2021; pp. 51–62. https://doi.org/10.18653/v1/2021.disrpt-1.6.
16. Kobayashi, N.; Hirao, T.; Kamigaito, H.; Okumura, M.; Nagata, M. A Simple and Strong Baseline for End-to-End Neural RST-style Discourse Parsing. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2022; Association for Computational Linguistics: Abu Dhabi, United Arab Emirates, 2022; pp. 6725–6737.
17. Tjong Kim Sang, E.F.; Déjean, H. Introduction to the CoNLL-2001 shared task: clause identification. In Proceedings of the Proceedings of the ACL 2001 Workshop on Computational Natural Language Learning (ConLL), 2001.
18. Marcus, M.P.; Santorini, B.; Marcinkiewicz, M.A. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* **1993**, *19*, 313–330.
19. Al-Thanyyan, S.S.; Azmi, A.M. Automated Text Simplification: A Survey. *ACM Comput. Surv.* **2021**, *54*. https://doi.org/10.1145/3442695.
20. Givón, T. *Syntax: An Introduction. Volume I*; John Benjamins, 2001.
21. Matthiessen, C.M. Combining clauses into clause complexes: A multi-faceted view. In *Complex Sentences in Grammar and Discourse*; John Benjamins, 2002; pp. 235–319.
22. Hopper, Paul J.; Traugott, E.C. *Grammaticalization*; Cambridge University Press, 2003.
23. Aarts, B. *Syntactic gradience: The nature of grammatical indeterminacy*; Oxford University Press, 2007.
24. Givón, T. *On Understanding Grammar: Revised edition*; John Benjamins, 2018; pp. 1–321.
25. Carter, R.; McCarthy, M. *Cambridge grammar of English: a comprehensive guide; spoken and written English grammar and usage*; Cambridge University Press, 2006.
26. Feng, S.; Banerjee, R.; Choi, Y. Characterizing Stylistic Elements in Syntactic Structure. In Proceedings of the Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning; Association for Computational Linguistics: Jeju Island, Korea, 2012; pp. 1522–1533.
27. Del Corro, L.; Gemulla, R. Clausie: clause-based open information extraction. In Proceedings of the Proceedings of the 22nd international conference on World Wide Web, 2013, pp. 355–366.
28. Vo, D.T.; Bagheri, E. Self-training on refined clause patterns for relation extraction. *Information Processing & Management* **2018**, *54*, 686–706. https://doi.org/https://doi.org/10.1016/j.ipm.2017.02.009.
29. Oberländer, L.A.M.; Klinger, R. Token Sequence Labeling vs. Clause Classification for English Emotion Stimulus Detection. In Proceedings of the Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics; Association for Computational Linguistics: Barcelona, Spain (Online), 2020; pp. 58–70.
30. Qi, P.; Zhang, Y.; Zhang, Y.; Bolton, J.; Manning, C.D. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In Proceedings of the Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations; Association for Computational Linguistics: Online, 2020; pp. 101–108. https://doi.org/10.18653/v1/2020.acl-demos.14.
31. Morey, M.; Muller, P.; Asher, N. How much progress have we made on RST discourse parsing? A replication study of recent results on the RST-DT. In Proceedings of the Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing; Association for Computational Linguistics: Copenhagen, Denmark, 2017; pp. 1319–1324. https://doi.org/10.18653/v1/D17-1136.
32. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* **2017**, *5*, 135–146. https://doi.org/10.1162/tacl_a_00051.
33. Dozat, T.; Manning, C.D. Deep Biaffine Attention for Neural Dependency Parsing. In Proceedings of the International Conference on Learning Representations, 2016, pp. 1–8.
34. Zeldes, A. The GUM Corpus: Creating Multilayer Resources in the Classroom. *Lang. Resour. Eval.* **2017**, *51*, 581–612. https://doi.org/10.1007/s10579-016-9343-x.

35. Asher, N.; Hunter, J.; Morey, M.; Farah, B.; Afantenos, S. Discourse Structure and Dialogue Acts in Multiparty Dialogue: the STAC Corpus. In Proceedings of the Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16); European Language Resources Association (ELRA): Portorož, Slovenia, 2016; pp. 2721–2727.
36. Heilman, M.; Sagae, K. Fast Rhetorical Structure Theory Discourse Parsing. *CoRR* **2015**, *abs/1505.02425*, [1505.02425].
37. Yu, Y.; Zhu, Y.; Liu, Y.; Liu, Y.; Peng, S.; Gong, M.; Zeldes, A. GumDrop at the DISRPT2019 Shared Task: A Model Stacking Approach to Discourse Unit Segmentation and Connective Detection. In Proceedings of the Proceedings of the Workshop on Discourse Relation Parsing and Treebanking 2019; Association for Computational Linguistics: Minneapolis, MN, 2019; pp. 133–143. https://doi.org/10.18653/v1/W19-2717.
38. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 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); Association for Computational Linguistics: Minneapolis, Minnesota, 2019; pp. 4171–4186. https://doi.org/10.18653/v1/N19-1423.
39. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach, 2019, [arXiv:cs.CL/1907.11692].
40. Joshi, M.; Chen, D.; Liu, Y.; Weld, D.S.; Zettlemoyer, L.; Levy, O. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics* **2020**, *8*, 64–77. https://doi.org/10.1162/tacl_a_00300.
41. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.R.; Le, Q.V. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In Proceedings of the Advances in Neural Information Processing Systems. Curran Associates, Inc., 2019, Vol. 32.
42. He, P.; Liu, X.; Gao, J.; Chen, W. DeBERTa: Decoding-enhanced BERT with Disentangled Attention, 2021, [arXiv:cs.CL/2006.03654].