

Article

Addressing Long-Distance Dependencies in AMR Parsing with Hierarchical Clause Annotation

Yunlong Fan ^{1,2} , Bin Li ^{1,2} , Yikemaiti Sataer ^{1,2}, Miao Gao ^{1,2}, Chuanqi Shi ^{1,2} 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, Ministry of Education, Southeast University, Nanjing 211189, China

* Correspondence: zqgao@seu.edu.cn

Abstract: Most natural language processing (NLP) tasks operationalize an input sentence as a sequence with token-level embeddings and features, despite its clausal structure. Taking abstract meaning representation (AMR) parsing as an example, recent parsers are empowered by transformers and pre-trained language models, but long-distance dependencies (LDDs) introduced by long sequences are still open problems. We argue that LDDs are not actually to blame for the sequence length but are essentially related to the internal clause hierarchy. Typically, non-verb words in a clause cannot depend on words outside of it, and verbs from different but related clauses have much longer dependencies than those in the same clause. With this intuition, we introduce a type of clausal feature, hierarchical clause annotation (HCA), into AMR parsing and propose two HCA-based approaches, HCA-based self-attention (HCA-SA) and HCA-based curriculum learning (HCA-CL), to integrate HCA trees of complex sentences for addressing LDDs. We conduct extensive experiments on two in-distribution (ID) AMR datasets (AMR 2.0 and AMR 3.0) and three out-of-distribution (OOD) ones (TLP, New3, and Bio). Experimental results show that our HCA-based approaches achieve significant and explainable improvements (0.7 Smatch score in both ID datasets; 2.3, 0.7, and 2.6 in three OOD datasets, respectively) against the baseline model and outperform the state-of-the-art (SOTA) model (0.7 Smatch score in the OOD dataset, Bio) when encountering sentences with complex clausal structures that introduce most LDD cases.

Keywords: hierarchical clause annotation; long-distance dependencies; AMR parsing; self-attention; curriculum learning



Citation: Fan, Y.; Li, B.; Sataer, Y.; Gao, M.; Shi, C.; Gao, Z. Addressing Long-Distance Dependencies in AMR Parsing with Hierarchical Clause Annotation. *Electronics* **2023**, *12*, 3908. <https://doi.org/10.3390/electronics12183908>

Academic Editor: Arkaitz Zubiaga

Received: 3 August 2023

Revised: 1 September 2023

Accepted: 14 September 2023

Published: 16 September 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Most natural language processing (NLP) tasks operationalize an input sentence as a word sequence with token-level embeddings and features, which creates long-distance dependencies (LDDs) when encountering long complex sentences such as dependency parsing [1], constituency parsing [2], semantic role labeling (SRL) [3], machine translation [4], discourse parsing [5], and text summary [6]. In previous works, the length of a sentence has been blamed for LDDs superficially, and several universal methods are proposed to cure this issue, e.g., hierarchical recurrent neural networks [7], long short-term memory (LSTM) [8], attention mechanism [9], Transformer [10], implicit graph neural networks [11], etc.

Abstract meaning representation (AMR) parsing [12], or translating a sentence to a directed acyclic semantic graph with relations among abstract concepts, has made strides in counteracting LDDs in different approaches. In terms of transition-based strategies, Peng et al. [13] propose a cache system to predict arcs between distant words. In graph-based methods, Cai and Lam [14] present a graph ↔ sequence iterative inference to overcome inherent defects of the one-pass prediction process in parsing long sentences.

In seq2seq-based approaches, Bevilacqua et al. [15] employ the Transformer-based pre-trained language model, BART [16], to address LDDs in long sentences. Among these categories, seq2seq-based approaches have become mainstream, and recent parsers [17–20] employ the seq2seq architecture with the popular codebase *SPRING* [15], achieving better performance. Notably, *HGAN* [20] integrates token-level features, syntactic dependencies (SDP), and SRL with heterogeneous graph neural networks and has become the state-of-the-art (SOTA) in terms of removing extra silver training data, graph-categorization, and ensemble methods.

However, these AMR parsers still suffer performance degradation when encountering long sentences with deeper AMR graphs [18,20] that introduce most LDD cases. We argue that the complexity of the clausal structure inside a sentence is the essence of LDDs, where clauses are the core units of grammar and center on a verb that determines the occurrences of other constituents [21]. Our intuition is that non-verb words in a clause typically cannot depend on words outside, while dependencies between verbs correspond to the inter-clause relations, resulting in LDDs across clauses [22].

To prove our claim, we demonstrate the AMR graph of a sentence from the AMR 2.0 dataset (<https://catalog.ldc.upenn.edu/LDC2017T10>, accessed on 11 June 2022) and distinguish the AMR relation distances in terms of different segment levels (clause/phrase/token) in Figure 1. Every AMR relation is represented as a dependent edge between two abstract AMR nodes that align to one or more input tokens. The dependency distances of inter-token relations are subtractive results from the indices of tokens aligned to the source and target nodes, while those of inter-phrase and inter-clause relations are calculated by indices of the headwords in phrases and the verbs in clauses, respectively. (The AMR relation distances between a main clause and a relative/appositive clause are decided by the modified noun phrase in the former and the verb in the latter). As can be observed:

- Dependency distances of inter-clause relations are typically much longer than those of inter-phrase and inter-token relations, leading to most of the LDD cases. For example, the AMR relation *anxious* $\xrightarrow{:ARG0-of}$ *go-01*, occurring in the clause “*I get very anxious*”, and its relative clause “*which does sort of go away . . .*”, has a dependency distance of 6 (subtracting the 9th token “*anxious*” from the 15th token “*go*”).
- Reentrant AMR nodes abstracted from pronouns also lead to far distant AMR relations. For example, the AMR relation *wait-01* $\xrightarrow{:ARG0}$ *I* has a dependency distance of 33 (subtracting the 1st token “*I*” from the 34th token “*wait*”).

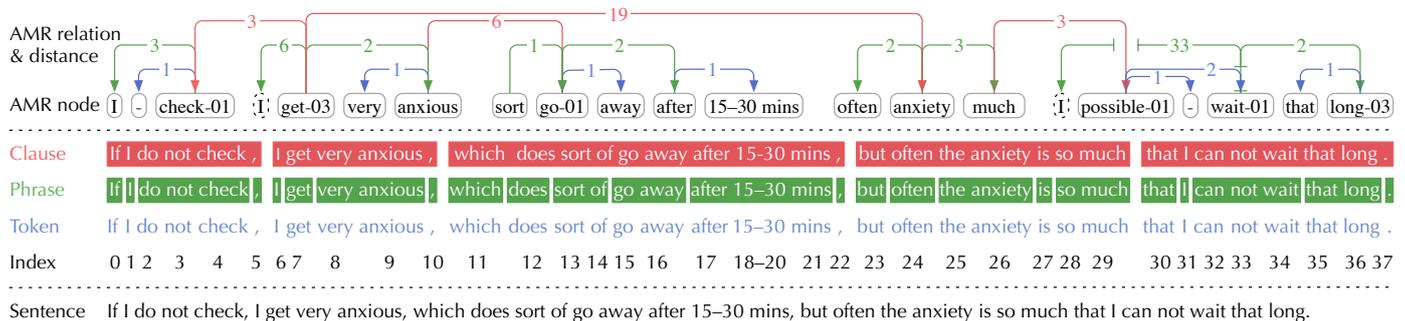


Figure 1. AMR relation dependency distances in different segment levels of an AMR 2.0 sentence. The input sentence is placed at the (bottom), and the sentence’s clause/phrase/token-level segments are positioned in the (middle) along with the token indices. The corresponding AMR graph is displayed at the (top), where AMR relations are represented as directed edges with a dependency distance, i.e., the indices’ subtraction of two tokens mapping to the source/target AMR nodes. Inter-clause/phrase/token relations are distinguished in separate colors, corresponding to the segment levels’ colors. Note that two virtual AMR nodes in dashed boxes of the reentrant node “*I*” are added for simplicity.

Based on the findings above, we are inspired to utilize the clausal features of a sentence to cure LDDs. Rhetorical structure theory (RST) [23] provides a general way to describe the coherence relations among clauses and some phrases, i.e., elementary discourse units, and postulates a hierarchical discourse structure called discourse tree. Except for RST, a novel clausal feature, hierarchical clause annotation (HCA) [24], also captures a tree structure of a complex sentence, where the leaves are segmented clauses and the edges are the inter-clause relations.

Due to the better parsing performances of the clausal structure [24], we select and integrate the HCA trees of complex sentences to cure LDDs in AMR parsing. Specifically, we propose two HCA-based approaches, HCA-based self-attention (HCA-SA) and HCA-based curriculum learning (HCA-CL), to integrate the HCA trees as clausal features in the popular AMR parsing codebase, *SPRING* [15]. In HCA-SA, we convert an HCA tree into a clause adjacency matrix and a token visibility matrix to restrict the attention scores between tokens from unrelated clauses and increase those from related clauses in masked-self-attention encoder layers. In HCA-CL, we employ curriculum learning with two training curricula, Clause-Number and Tree-Depth, with the assumption that “the more clauses or the deeper clausal tree in a sentence, the more difficult it is to learn”.

We conduct extensive experiments on two in-distribution (ID) AMR datasets (i.e., AMR 2.0 and AMR 3.0 (<https://catalog.ldc.upenn.edu/LDC2020T02>, accessed on 11 June 2022)) and three out-of-distribution (OOD) ones (i.e., TLP, New3, and Bio) to evaluate our two HCA-based approaches. In ID datasets, our parser achieves a 0.7 Smatch F1 score improvement against the baseline model, *SPRING*, on both AMR 2.0 and AMR 3.0, and outperforms the SOTA parser, *HGAN*, by 0.5 and 0.6 F1 scores for the fine-grained metric *SRL* in the two datasets. Notably, as the clause number of the sentence increases, our parser outperforms *SPRING* by a large margin and achieves better Smatch F1 scores than *HGAN*, indicating the ability to cure LDDs. In OOD datasets, the performance boosts achieved by our HCA-based approaches are more evident in complicated corpora like New3 and Bio, where sentences consist of more clauses and longer clauses. Our code is publicly available at <https://github.com/MetroVancloud/HCA-AMRparsing> (accessed on 3 August 2023).

The rest of this paper is organized as follows. The related works are summarized in Section 2, and the proposed approaches are detailed in Section 3. Then, the experiments of AMR parsing are presented in Section 4. Next, the discussion of the experimental results is presented in Section 5. Finally, our work is concluded in Section 6.

2. Related Work

In this section, we first introduce the open problem of LDDs and some universal methods. Then, we summarize the four main categories of parsers and the LDD cases in AMR parsing. Finally, we introduce the novel clausal feature, HCA.

2.1. Long-Distance Dependencies

LDDs, first proposed by Hockett [25], describe an interaction between two (or more) elements in a sequence separated by an arbitrary number of positions. LDDs are related to the rate of decay of statistical dependence of two points with increasing time intervals or spatial distances between them [26]. In recent linguistic research into LDDs, Liu et al. [22] propose the verb-frame frequency account to robustly predict acceptability ratings in sentences with LDDs, indicating the affinities between the number of verbs and LDDs in sentences.

In recent advances in NLP tasks, hierarchical recurrent neural networks [7], LSTM [8], attention mechanism [9], Transformer [10], and implicit graph neural networks [11] are proposed to cure LDDs. Specifically, the attention mechanism has been showcased with successful applications to address LDDs in diverse environments. For instance, Xiong and Li [27] designed an attention mechanism to enable the neural model to learn relevant and informative words that contain topic-related information in students’ constructed response answers. It solved long-distance dependencies by focusing on specific parts of the input

sequence that are most relevant to the task at hand. Zukov-Gregoric et al. [28] proposed a self-attention mechanism in the multi-head encoder–decoder neural network architecture that allows the network to focus on important resolution information from long writings, enabling it to perform better in named entity recognition. Li et al. [29] also employed a bidirectional LSTM model with a self-attention mechanism to enhance the sentiment information derived from existing linguistic knowledge and sentiment resources in the sentiment analysis task.

These universal neural models all represent the input sequence with token-level embeddings from pretrained language models in most NLP tasks.

2.2. AMR Parsing

AMR parsing is a challenging semantic parsing task since AMR is a deep semantic representation consisting of many special annotations (e.g., abstract concept nodes, named entities, co-references, and such) [12]. The aim of AMR parsing is translating a sentence to a directed acyclic semantic graph with relations among abstract concepts, where the two main characteristics are:

1. *Abstraction*: Assigns the same AMR to sentences with the same basic meaning and also brings a challenge for alignments between input tokens and output AMR nodes [12], e.g., the token “can” and its corresponding AMR node *possible-01* in Figure 1.
2. *Reentrancy*: Introduces the presence of nodes with multiple parents and represents sentences as graphs rather than trees [30], causing some LDD cases, e.g., the AMR node “I” in Figure 1.

Existing AMR parsers can be summarized into four categories:

1. *Graph-based*: Directly predict nodes and edges in either two-stage procedures [31–33] or incremental one-stage [14,34] procedures. The SOTA graph-based model, *AMR-gs* [14], enhances the incremental graph construction with an AMR graph \leftrightarrow sequence iterative inference mechanism in one-stage procedures.
2. *Seq2seq-based*: Model the task as transduction of the sentence into a linearization of the AMR graph [15,17–20,35–38]. *SPRING* [15] is a popular seq2seq-based codebase that employs transfer learning by exploiting a pretrained encoder–decoder model, BART [16], to generate a linearized graph incrementally with a single auto-regressive pass of a seq2seq decoder. The subsequent models, *ANCES* [17], *HCL* [18], *ATP* [19], and *HGAN* [20] all follow the architecture of *SPRING*, and *HGAN* integrates SDP and SRL features with heterogeneous graph neural networks to achieve the best performance in the tasks of removing extra silver training data, graph re-categorization, and ensemble methods.
3. *Transition-based*: Predict a sequence of actions that generate the graph while processing tokens left-to-right through the sentence [39–45]. The SOTA transition-based model, *StructBART* [45], explores the integration of general pre-trained sequence-to-sequence language models and a structure-aware transition-based approach.
4. *Grammar-based*: Peng et al. [46] introduce a synchronous hyperedge replacement grammar solution. Pust et al. [47] regard the task as a machine translation problem, while Artzi et al. [48] adapt combinatory categorical grammar. Groschwitz et al. [49] and Lindemann et al. [50] view AMR graphs as the structural AM algebra.

Despite different architectures, most AMR parsers employ word embeddings from pretrained language models and utilize token-level features like part-of-speech (POS), SDP, and SRL. However, these parsers still suffer performance degradation [18,20] when parsing complex sentences with LDDs due to the difficulties of aligning input tokens with output AMR nodes in such a long sequence.

2.3. Hierarchical Clause Annotation

RST [23] provides a general way to describe the coherence relations among parts in a text and postulates a hierarchical discourse structure called a discourse tree. The leaves of

a discourse tree can be a clause or a phrase without strict definitions, known as elementary discourse units. However, the performances of RST parsing tasks are unsatisfactory due to the loose definitions of elementary discourse units and abundant types of discourse relations [51,52].

Syntactic dependency parse trees (SDPT) and constituency parse trees (CPT) are existing syntactic representations that provide hierarchical token-level annotations. However, AMR has the unique feature, *abstraction*, summarized in Section 2.2, indicating the challenge of alignments between input tokens and output AMR nodes. When encountering long and complex sentences, the performance of parsing SDPT [1] and CPT [2] degrade, and these silver token-level annotations also contribute more noise for “token-node” alignments in AMR parsing.

In addition to RST, Fan et al. [24] also propose a novel clausal feature, HCA, which represents a complex sentence as a tree consisting of clause nodes and inter-clause relation edges. The HCA framework is based on English grammar [21], where clauses are elementary grammar units that center around a verb, and inter-clause relations can be classified into two categories:

1. Coordination: An equal relation shared by clauses with the same syntactic status, including *And*, *Or*, and *But* relations.
2. Subordination: Occurs in a matrix and a subordinate clause, including *Subjective*, *Objective*, *Predicative*, *Appositive*, *Relative*, and nine sublevel *Adverbial* relations.

Inter-clause relations have appropriate alignments with AMR relations, where nominal clause relations correspond to the frame arguments (e.g., *Subjective* vs. :ARG0), and adverbial clause relations are mapped to general semantic relations (e.g., *Adverbial_of_Condition* vs. :condition). Figure 2 demonstrates the segmented clauses and the HCA tree of the same sentence in Figure 1.

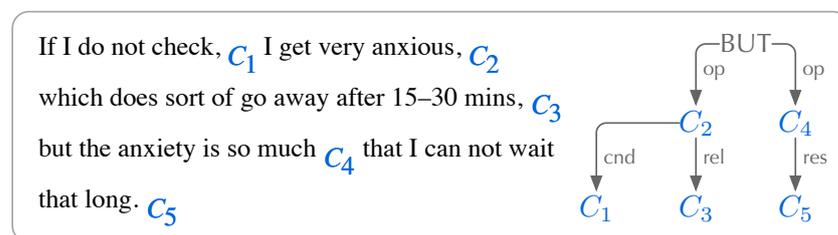


Figure 2. Segmented clauses and the HCA tree of a sentence in AMR 2.0. Clauses C₂ and C₄ are contrasted and coordinated, dominated by the node *BUT*. Clauses C₁, C₃, and C₅ are subordinated to their matrix clauses, where *cnd*, *rel*, and *res* represent the inter-clause relations of *Adverbial_of_Condition*, *Relative*, and *Adverbial_of_Result*, respectively.

Based on well-defined clauses and inter-clause relations, Fan et al. provide a manually annotated HCA corpus for AMR 2.0 and high-performance neural models to generate silver HCA trees for more complex sentences. Therefore, we select and utilize the HCA trees as clausal features to address LDDs in AMR parsing.

3. Model

In this paper, we propose two HCA-based approaches, HCA-SA (Section 3.1) and HCA-CL (Section 3.2), to integrate HCA trees in the popular AMR parser codebase *SPRING* for addressing LDDs in AMR parsing.

3.1. HCA-Based Self-Attention

Existing AMR parsers (e.g., *SPRING*) employ Transformer [10] as an encoder to obtain the input sentence representation. However, in the standard self-attention mechanism adopted by the Transformer model, every token needs to attend to all other tokens, and the learned attention matrix *A* is often very sparse across most data points [53]. Inspired by the work of Liu et al. [54], we propose the HCA-SA approach, which utilizes hierarchical

clause annotations as a structural bias to restrict the attention scope and attention scores. We summarize this method in Algorithm 1.

Algorithm 1 HCA-Based Self-Attention

Require: Attention head number h , attention matrices Q, K , and V , token visibility matrices M^{vis} and M^{deg} , matrix that maps the attention head indices to clause relations M^{rel}
Ensure: Multi-head attention weights $A^{\text{MultiHead}}$ with embedded HCA features

- 1: Initialization: Multi-head attention weights $A^{\text{MultiHead}}$, attention layer index $i \leftarrow 0$
- 2: **repeat**
- 3: $M_i^{\text{vis}} \leftarrow M^{\text{vis}} M_i^{\text{rel}}$ \triangleright select a certain type of clause relation of this attention head
- 4: $M_i^{\text{deg}} \leftarrow M^{\text{vis}} M_i^{\text{rel}}$
- 5: $S_i^{\text{mask}} \leftarrow \text{Softmax}(\frac{QK^T}{\sqrt{d}} M_i^{\text{deg}} + M_i^{\text{vis}})$ \triangleright mask the attention scores with HCA
- 6: $A_i^{\text{mask}} \leftarrow S_i^{\text{mask}} V$
- 7: $A^{\text{MultiHead}} \leftarrow \text{Concat}(A^{\text{MultiHead}}, A_i^{\text{mask}})$ \triangleright concatenate each attention weight
- 8: $i \leftarrow i + 1$
- 9: **until** $i = h$

3.1.1. Token Visibility Matrix

For the example sentence in Figure 2, its HCA tree can be transferred into a clause adjacency matrix in Figure 3a by checking if two clauses have an inter-clause edge (not meaning that they are adjacent in the source sentence), where adjacent clauses have specific correlations (pink) and non-adjacent ones share no semantics (white). Each clause has the strongest correlation with itself (red).

Furthermore, we transform the clause adjacency matrix into token visibility matrices by splitting every clause into tokens. As shown in Figure 3b,c, the visibility between tokens can be summarized into the following cases:

- *Full Visibility:* tokens in the same clause are fully mutually visible;
- *Partial Visibility:* tokens from two adjacent clauses C_1 and C_2 share partial visibility;
- *No Visibility:* tokens from non-adjacent clauses C_2 and C_5 are invisible to each other;
- *Global Visibility:* tokens with a pronoun POS (e.g., “I” in C_5) are globally visible for the linguistic phenomena of co-reference;
- *Additional Visibility:* tokens that are clausal keywords (i.e., coordinators, subordinators, and antecedents) share additional visibilities with the tokens in adjacent clauses (e.g., “if” in C_1 to tokens in C_2).

Therefore, we introduce two token visibility matrices M^{vis} and M^{deg} , where the former signals whether two tokens are *visible* mutually, and the latter measures the visibility *degree* between them:

$$M_{i,j}^{\text{vis}} = \begin{cases} -\infty & w_i \otimes w_j \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$M_{i,j}^{\text{deg}} = \begin{cases} 0 & w_i \otimes w_j \\ 1, & w_i \odot w_j \\ \lambda_0, & w_i \ominus w_j \\ \lambda_1, & w_i \ominus w_j \wedge \text{Key}(w_i/w_j) \\ \mu, & \text{PRN}(w_i/w_j), \end{cases} \tag{2}$$

where \otimes, \odot , and \ominus mean that the corresponding two tokens, w_i and w_j , are positioned in two nonadjacent (*No Visibility*), similar (*Full Visibility*), and adjacent (*Partial Visibility*) clauses, respectively. $\text{Key}(w_i/w_j)$ indicates that at least one of w_i and w_j is a clausal keyword token, while $\text{PRN}(w_i/w_j)$ denotes the existence of at least one pronoun in w_i and w_j . Values of the hyperparameters λ_0, λ_1 , and μ are in $(0, 1)$, and $\lambda_1 > \lambda_0$.

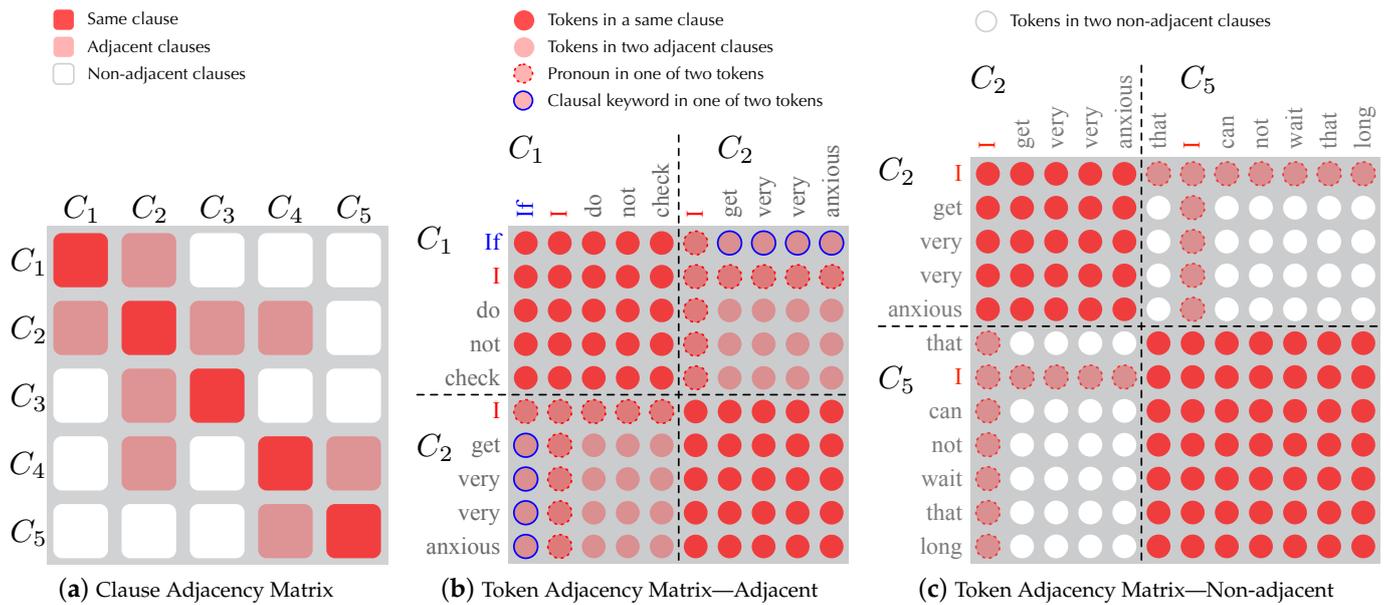


Figure 3. Overview of our hierarchical clause annotation (HCA)-based self-attention approach that integrates the clausal structure of input sentences. In (a), red blocks mean that clauses have the strongest correlation with themselves; the pink/white ones mean that the corresponding two are adjacent/non-adjacent in the HCA tree. In (b,c), the adjacency between two clauses is concretized in a token visibility matrix. Pink circles with a red dotted border mean one of the two corresponding tokens is a pronoun, while those with a blue solid border indicate the existence of a clausal keyword (i.e., coordinator, subordinator, or antecedent).

3.1.2. Masked-Self-Attention

To some degree, the token visibility matrices M^{vis} and M^{deg} contain the structural information of the HCA tree. For vanilla transformers employed in existing AMR parsers, stacked self-attention layers inside can not receive M^{vis} and M^{deg} as inputs, so we modify this to *Masked-Self-Attention*, which can restrict the attention scope and attention scores according to M^{vis} and M^{deg} . Formally, the masked attention scores S^{mask} and the masked attention matrix A^{mask} are defined as:

$$S^{mask} = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}M^{deg} + M^{vis}\right) \tag{3}$$

$$A^{mask} = S^{mask}V, \tag{4}$$

where self-attention inputs are $Q, K, V \in \mathbb{R}^{N \times d}$; N is the length of the input sentences; and scaling factor d is the dimension of the model. Intuitively, if token w_i is invisible to w_j , the attention scores $S_{i,j}^{mask}$ will be masked to 0 due to the value $-\infty$ of $M_{i,j}^{vis}$ and the value 0 of $M_{i,j}^{deg}$. Otherwise, $S_{i,j}^{mask}$ will be scaled according to $M_{i,j}^{deg}$ in different cases.

3.1.3. Clause-Relation-Bound Attention Head

In every stacked self-attention layer of Transformer, multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions [10]. It also provides us the possibility of integrating different inter-clause relations in different attention heads. Instead of masking the attention matrix with non-labeled HCA trees, we propose a clause-relation-bound attention head setting, where every head attends to a specific inter-clause relation.

In this setting, we increase the visibility between tokens in two adjacent clauses with interrelation rel_i in the attention matrix $A_{rel_i}^{mask}$ of the bound head, i.e., increasing λ_0 to 1 in

Equation (2). Therefore, the final attention matrix $A^{\text{MultiHead}}$ of each stacked self-attention layer is defined as:

$$A^{\text{MultiHead}} = \text{Concat}(A_{\text{rel}_1}^{\text{mask}}, \dots, A_{\text{rel}_i}^{\text{mask}}, \dots, A_{\text{rel}_h}^{\text{mask}})W^O, \tag{5}$$

where the parameter matrix $W^O \in \mathbb{R}^{hN \times d}$, and h is the attention head number mapping to 16 inter-clause relations in HCA [24].

3.2. HCA-Based Curriculum Learning

Inspired by the idea of curriculum learning [55], which suggests that humans handle difficult tasks from easy examples to hard ones, we propose an HCA-CL approach for training an AMR parser, in which the clause number and the tree depth of a sentence’s HCA are the measurements of learning difficulty. Referring to the previous work of Wang et al. [18], we set two learning curricula, *Clause-Number* and *Tree-Depth*, in our HCA-CL approach demonstrated in Figure 4.

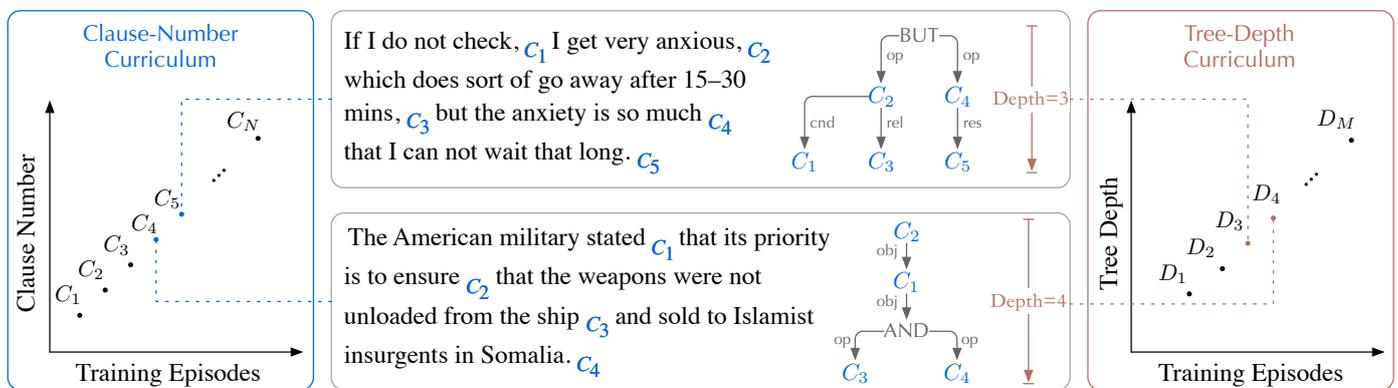


Figure 4. Overview of our hierarchical clause annotation (HCA)-based curriculum learning approach with two curricula, *Clause-Number* and *Tree-Depth*. The learning difficulties of the two curricula are set by the clause number and the tree depth of a sentence’s HCA in the (left) and (right) charts. Two example sentences from AMR 2.0 and their HCAs are demonstrated in the middle.

3.2.1. Clause-Number Curriculum

In the *Clause-Number* (CN) curriculum, sentences with more clauses that involve more inter-clause relations and longer dependency distances (demonstrated in Figure 1) are considered to be harder to learn. Given this assumption, all training sentences are divided into N buckets according to their clause number $\{C_i : i = 1, \dots, N\}$, where C_i contains sentences with the clause number i . In the training epoch of the *Clause-Number* curriculum, there are N training episodes with T_{cn} steps individually. At each step of the i -th episode, the training scheduler samples a batch of examples from buckets $\{I_j : j \leq i\}$ to train the model.

3.2.2. Tree-Depth Curriculum

In the *Tree-Depth* (TD) curriculum, sentences with deeper HCA trees that correspond to deeper hierarchical AMR graphs are considered to be harder to learn. Given this assumption, all training sentences are divided into M buckets according to their clause number $\{D_i : i = 1, \dots, M\}$, where D_i contains sentences with the clause number i . In the training epoch of the *Clause-Number* curriculum, there are M training episodes with T_{td} steps individually. At each step of the i -th episode, the training scheduler samples a batch of examples from buckets $\{I_j : j \leq i\}$ to train the model.

4. Experiments

In this section, we describe the details of datasets, environments, model hyperparameters, evaluation metrics, compared models, and parsing results for the experiments.

4.1. Datasets

For the benchmark datasets, we choose two standard AMR datasets, AMR 2.0 and AMR 3.0, as the ID settings and three test sets, TLP, New3, and Bio, as the OOD settings.

For the HCA tree of each sentence, we use the manually annotated HCA trees for AMR 2.0 provided by [24] and auto-annotated HCA trees for the remaining datasets, which were all generated by the HCA Segmenter and the HCA Parser proposed by [24].

4.1.1. In-Distribution Datasets

We first train and evaluate our HCA-based parser on two standard AMR parsing evaluation benchmarks:

- *AMR 2.0* includes 39,260 sentence–AMR pairs in which source sentences are collected from: the DARPA BOLT and DEFT programs, transcripts and English translations of Mandarin Chinese broadcast news programming from China Central TV, text from the Wall Street Journal, translated Xinhua news texts, various newswire data from NIST OpenMT evaluations, and weblog data used in the DARPA GALE program.
- *AMR 3.0* is a superset of AMR 2.0 and enriches the data instances to 59,255. New source data added to AMR 3.0 include sentences from Aesop’s Fables, parallel text and the situation frame dataset developed by LDC for the DARPA LORELEI program, and lead sentences from Wikipedia articles about named entities.

The training, development, and test sets in both datasets are a random split, and therefore we take them as ID datasets as in previous works [15,17,18,20,45].

4.1.2. Out-of-Distribution Datasets

To further estimate the effects of our HCA-based approaches on open-world data that come from a different distribution, we follow the OOD settings introduced by [15] and predict based on three OOD test sets with the parser trained on the AMR 2.0 training set:

- *New3* (Accessed at <https://catalog.ldc.upenn.edu/LDC2020T02>, accessed on 11 August 2022): A set of 527 instances from AMR 3.0, whose source was the LORELEI DARPA project (not included in the AMR 2.0 training set) consisting of excerpts from newswire and online forums.
- *TLP* (Accessed at <https://amr.isi.edu/download/amr-bank-struct-v3.0.txt>, accessed on 12 October 2022; note that the annotators did not mention the translators of this English version applied in the annotation): The full AMR-tagged children’s novel written by Antoine de Saint-Exupéry, *The Little Prince* (version 3.0), consisting of 1562 pairs.
- *Bio* (<https://amr.isi.edu/download/2016-03-14/amr-release-test-bio.txt>, accessed on 15 October 2022): the test set of the Bio-AMR corpus, consisting of 500 instances, featuring biomedical texts [56].

4.1.3. Hierarchical Clause Annotations

For the hierarchical clausal features utilized in our HCA-based approaches, we use the manually annotated HCA corpus for AMR 2.0 provided in [24]. Moreover, we employ the HCA segmenter and the HCA parser proposed in [24] to generate silver HCA trees for AMR 3.0 and three OOD test sets. Detailed statistics of the evaluation datasets in this paper are listed in Table 1.

Table 1. Main statistics of five AMR parsing benchmarks. “ID” and “OOD” denote in-distribution and out-of-distribution settings, respectively. “#Snt.” and “#HCA” represent the total number of sentences and complex sentences with hierarchical clause annotations in each split set.

	Dataset	Training		Development		Test	
		#Snt.	#HCA	#Snt.	#HCA	#Snt.	#HCA
ID	AMR 2.0	36,521	17,886	1368	741	1371	753
	AMR 3.0	55,635	36,921	1722	1243	1898	1258
OOD	New3	-	-	-	-	527	286
	TLP	-	-	-	-	1562	825
	Bio	-	-	-	-	500	367

4.2. Baseline and Compared Models

We compare our HCA-based AMR parser with several recent parsers:

- *AMR-gs (2020)* [14], a graph-based parser that enhances incremental graph construction with an AMR graph \leftrightarrow sequence (*AMR-gs*) iterative inference mechanism in one-stage procedures.
- *APT (2021)* [44], a transition-based parser that employs an action-pointer transformer (APT) to decouple source tokens from node representations and address alignments.
- *StructBART (2021)* [45], a transition-based parser that integrates the pre-trained language model, BART, for structured fine-tuning.
- *SPRING (2021)* [15], a fine-tuned BART model that predicts a linearized AMR graph.
- *HCL (2022)* [18], a hierarchical curriculum learning (*HCL*) framework that helps the seq2seq model adapt to the AMR hierarchy.
- *ANCES (2022)* [17], a seq2seq-based parser that adds the important ancestor (*ANCES*) information into the Transformer decoder.
- *HGAN (2022)* [20], a seq2seq-based parser that applies a heterogeneous graph attention network (*HGAN*) to argument word representations with syntactic dependencies and semantic role labeling of input sentences. It is also the current SOTA parser in the settings of removing graph re-categorization, extra silver training data, and ensemble methods.

Since *SPRING* provides a clear and efficient seq2seq-based architecture based on a vanilla BART, recent seq2seq-based models *HCA*, *ANCES*, and *HGAN* all select it as the codebase. Therefore, we also choose *SRPING* as the baseline model to apply our HCA-based approaches. Additionally, we do not take the competitive AMR parser, *ATP* [19], into consideration for our compared models since it employs syntactic dependency parsing and semantic role labeling as intermediate tasks to introduce extra silver training data.

4.3. Hyper-Parameters

For the hyper-parameters of our HCA-based approaches, we list their layer, name, and value in Table 2. To pick the hyper-parameters employed in the HCA-SA Encoder, i.e., λ_0 , λ_1 , and μ , we use a random search with a total of 16 trials in their search spaces (λ_0 : [0.1, 0.6], λ_1 : [0.4, 0.9], μ : [0.7, 1.0], and stride +0.1). According to the results of these experimental trials, we selected the final pick for each hyper-parameter. All models are trained until reaching their maximum epochs, and then we select the best model checkpoint on the development set.

Table 2. Final hyper-parameter configuration of the clause segmentation model. “HCA-SA Encoder” indicates the HCA-based self-attention approach user in the encoder, and “HCA-CL Strategy” represents the HCA-based curriculum learning approach used before the normal training epochs.

Layer	Hyper-Parameter	Value
Word Embedding	BART-large	1024
HCA-SA Encoder	layer	12
	head	16
	λ_0	0.5
	λ_1	0.8
	μ	1
Decoder	layer	12
	head	16
HCA-CL Strategy	T_{cn}	500
	T_{td}	1500
Trainer	optimizer	RAdam
	weight decay	4×10^{-3}
	loss function	Cross-entropy
	learning rate	5×10^{-5}
	batch size	500
	dropout	0.25
	maximum epochs	30
Prediction	beam size	5

4.4. Evaluation Metrics

Following previous AMR parsing works, we use Smatch scores [57] and fine-grained metrics [58] to evaluate the performances. Specifically, the fine-grained AMR metrics are:

1. Unlabeled (*Unlab.*): Smatch computed on the predicted graphs after removing all edge labels.
2. No WSD (*NoWSD*): Smatch while ignoring Propbank senses (e.g., “go-01” vs. “go-02”).
3. Named entity (*NER*): F-score on the named entity recognition (:name roles).
4. Wikification (*Wiki.*): F-score on the wikification (:wiki roles).
5. Negation (*Neg.*): F-score on the negation detection (:polarity roles).
6. Concepts (*Conc.*): F-score on the concept identification task.
7. Reentrancy (*Reent.*): Smatch computed on reentrant edges only, e.g., the edges of node “I” in Figure 1.
8. Semantic role labelings (*SRL*): Smatch computed on :ARGi roles only.

As suggested in [18], *Unlab.*, *Reent.*, and *SRL* are considered to be structure-dependent metrics, since:

- *Unlab.* does not consider any edge labels and only considers the graph structure;
- *Reent.* is a typical structure feature for the AMR graph. Without reentrant edges, the AMR graph is reduced to a tree;
- *SRL* denotes the core-semantic relation of the AMR, which determines the core structure of the AMR.

Conversely, all other metrics are classified as structure-independent metrics.

4.5. Experimental Environments

Table 3 lists the information on the main hardware and software used in our experimental environments. Note that the model in AMR 2.0 is trained for a total of 30 epochs for 16 h, while the model trained in AMR 3.0 is finished in a total of 30 epochs for 28 h given the experimental environments.

Table 3. Hardware and software used in our experiments.

Environment	Value
Hardware	
CPU	Intel(R) Xeon(R) Silver 4216 CPU @ 2.10 GHz
GPU	NVIDIA RTX 2080Ti (11 G)
Memory	64 GB
Software	
Python	3.8.16
Pytorch	1.13.0
Anaconda	4.10.1
CUDA	11.0
IDE	PyCharm 2022.2.3

4.6. Experimental Results

We now report the AMR parsing performances of our HCA-based parser and other comparison parsers on ID datasets and OOD datasets, respectively.

4.6.1. Results in ID Datasets

As demonstrated in Table 4, we report AMR parsing performances of the baseline model (*SPRING*), other compared parsers, and the modified *SPRING* that applies our HCA-based self-attention (*HCA-SA*) and curriculum learning (*HCA-CL*) approaches on ID datasets AMR 2.0 and AMR 3.0. All the results of our HCA-based model have averaged scores of five experimental trials, and we compute the significance of performance differences using the non-parametric approximate randomization test [59]. From the results, we can make the following observations:

- Equipped with our *HCA-SA* and *HCA-CL* approaches, the baseline model *SPRING* achieves a 0.7 Smatch F1 score improvement on both AMR 2.0 and AMR 3.0. The improvements are significant, with $p < 0.005$ and $p < 0.001$, respectively.
- In AMR 2.0, our HCA-based model outperforms all compared models except *ANCES* and the *HGAN* version that introduces both DP and SRL features.
- In AMR 3.0, consisting of more sentences with HCA trees, the performance gap between our HCA-based parser and the SOTA (*HGAN* with DP and SRL) is only a 0.2 Smatch F1 score.

To better analyze how the performance improvements of the baseline model are achieved when applying our HCA-based approaches, we also report structure-dependent fine-grained results in Table 4. As claimed in Section 1, inter-clause relations in the HCA can bring LDD issues, which are typically related to AMR concept nodes aligned with verb phrases and reflected in structure-dependent metrics. As can be observed:

- Our HCA-based model outperforms the baseline model in nearly all fine-grained metrics, especially in structure-dependent metrics, with 1.1, 1.8, and 3.9 F1 score improvements in *Unlab.*, *Reent.*, and *SRL*, respectively.
- In the *SRL*, *Conc.*, and *Neg* metrics, our HCA-based model achieves the best performance against all compared models.

4.6.2. Results in OOD Datasets

As demonstrated in Table 5, we report the parsing performances of our HCA-based model and compared models on three OOD datasets. As can be seen:

- Our HCA-based model outperforms the baseline model *SPRING* with 2.5, 0.7, and 3.1 Smatch F1 score improvements in the New3, TLP, and Bio test sets, respectively.
- In the New3 and Bio datasets that contain long sentences of newswire and biomedical texts and have more HCA trees, our HCA-based model outperforms all compared models.

- In the TLP dataset that contains many simple sentences of a children’s story and fewer HCA trees, our HCA-based does not perform as well as *HCL* and *HGAN*.

Table 4. Smatch and fine-grained F1 scores (%) of our AMR parser and comparative ones on two in-distribution (ID) evaluation test sets. The column “Feat.” means the extra features that an AMR parser requires, where “DP”, “SRL”, and “HCA” indicate syntactic dependencies, semantic role labelings, and hierarchical clause annotations, respectively. For comparison fairness, the performances of compared parsers are the versions without graph re-categorization, extra silver training data, and ensemble methods. The best result per measure across each test set is shown in bold, while that in the baseline model (*SPRING*) and ours is underlined. “w/o” denotes “without”.

	Model	Feat.	Smatch	Structure-Dependent			Structure-Independent				
				Unlab.	Reent.	SRL	NoWSD	Conc.	Wiki.	NER	Neg.
AMR 2.0	AMR-gs (2020) [14]	-	78.7	81.5	63.8	74.5	79.2	88.1	81.3	87.1	66.1
	APT (2021) [44]	-	81.7	85.5	71.1	80.8	82.3	88.7	78.8	88.5	69.7
	StructBART (2021) [45]	-	84.3	87.9	74.3	-	-	-	-	-	-
	HCL (2022) [18]	-	84.3	87.7	74.5	83.2	85.0	90.2	84.0	91.6	75.9
	ANCES (2022) [17]	-	84.8	88.1	75.1	83.4	85.3	90.5	84.1	91.8	74.0
	HGAN (2022) [20]	DP	84.4	-	-	-	-	-	-	-	-
	HGAN (2022) [20]	DP, SRL	84.9	87.8	73.9	83.0	85.5	90.8	84.6	91.9	74.7
	SPRING (2021) [15]	-	83.8	86.1	70.8	79.6	84.4	90.2	84.3	90.6	74.4
	Ours	HCA	<u>84.5</u>	<u>87.0</u>	<u>72.5</u>	<u>83.2</u>	<u>84.5</u>	<u>90.7</u>	<u>84.4</u>	<u>91.2</u>	<u>75.2</u>
	AMR 3.0	AMR-gs (2020) [14]	-	78.0	81.9	63.7	73.2	78.5	88.5	75.7	83.7
APT (2021) [44]		-	80.3	-	-	-	-	-	-	-	-
StructBART (2021) [45]		-	83.2	-	-	-	-	-	-	-	-
HCL (2022) [18]		-	83.7	86.9	73.9	82.4	84.2	89.5	82.6	89.0	73.0
ANCES (2022) [17]		-	83.5	86.6	74.2	82.2	84.0	89.5	81.5	88.9	72.6
HGAN (2022) [20]		DP	83.5	-	-	-	-	-	-	-	-
HGAN (2022) [20]		DP, SRL	83.9	86.5	73.0	82.2	84.3	90.2	83.0	89.2	73.2
SPRING (2021) [15]		-	83.0	85.4	70.4	78.9	<u>83.5</u>	89.8	82.7	87.2	73.0
Ours		HCA	<u>83.7</u>	<u>86.6</u>	<u>72.2</u>	<u>82.2</u>	83.4	90.5	<u>82.6</u>	<u>88.0</u>	<u>73.8</u>

Table 5. Smatch F1 scores (%) of our HCA-based model and comparison models on out-of-distribution (OOD) datasets. The best result for each test set is shown in bold.

	New3	TLP	Bio
SPRING (2022) [15]	73.7	77.3	59.7
HCL (2022) [18]	75.3	78.2	61.1
HGAN (2022) [20]	76.0	79.2	61.6
Ours	76.0	78.0	62.3

5. Discussion

As shown in the previous section, our HCA-based model achieves prominent improvements against the baseline model, *SPRING*, and outperforms other compared models, including the SOTA model *HGAN* in some fine-grained metrics in ID and ODD datasets. In this section, we further discuss the paper’s main issue of whether our HCA-based approaches have any effect on curing LDDs. Additionally, the ablation studies and the case studies are also provided.

5.1. Effects on Long-Distance Dependencies in ID Datasets

As claimed in Section 1, most LDD cases occur in sentences with complex hierarchical clause structures. In Figure 5, we demonstrate the parsing performance trends of the baseline model *SPRING*, the SOTA parser *HGAN* (we only use the original data published in their paper to draw performance trends over the number of tokens, without performances

in terms of the number of clauses), and our HCA-based model over the number of tokens and clauses in sentences from AMR 2.0. It can be observed that:

- When the number of tokens (denoted as $\#Token$ for simplicity) >20 in a sentence, the performance boost of our HCA-based model against the baseline *SPRING* gradually becomes significant.
- For the case $\#Token > 50$ that indicates sentences with many clauses and inter-clause relations, our HCA-based model outperforms both *SPRING* and *HGAN*.
- When compared with performances trends for $\#Clause$, the performance lead of our HCA-based model against *SPRING* becomes much more evident as $\#Clause$ increases.

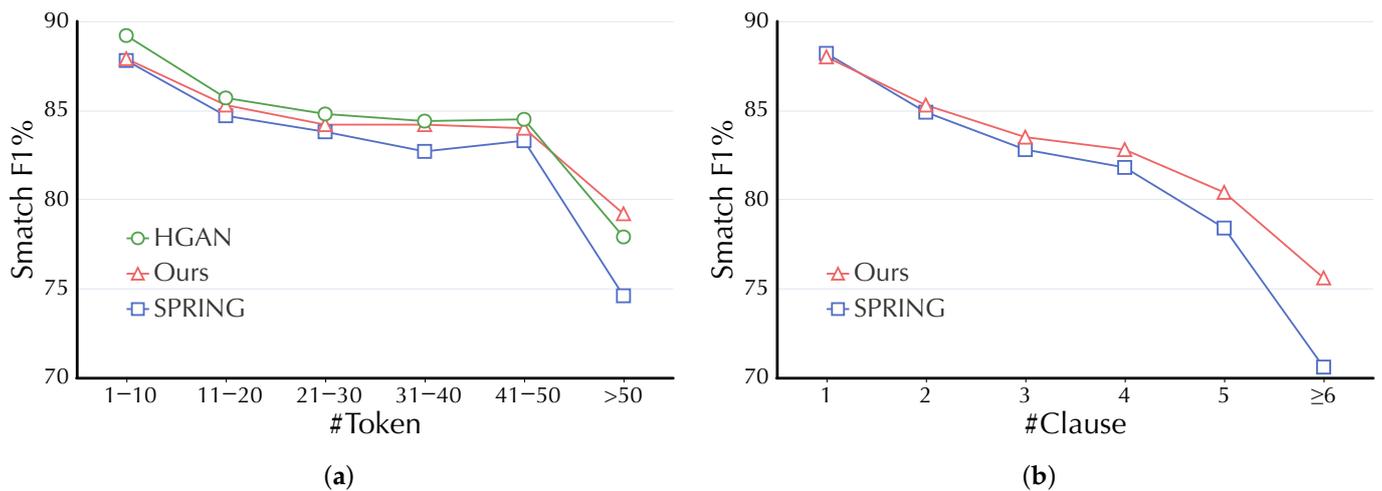


Figure 5. Performances trends of *SPRING*, *HGAN*, and *Ours* in the AMR 2.0 dataset over the number of tokens (denoted as “ $\#Token$ ”) and clauses (denoted as “ $\#Clause$ ”) inside a sentence. (a) Performance trends over the number of tokens. (b) Performance trends over the number of clauses.

To summarize, our HCA-based approaches show significant effectiveness on long sentences with complex clausal structures that introduce most LDD cases.

5.2. Effects on Long-Distance Dependencies in OOD Datasets

As the performance improvements achieved by our HCA-based approaches are much more prominent in OOD datasets than in ID datasets, we further explore the OOD datasets with different characteristics.

Figure 6 demonstrates the two main statistics of three OOD datasets, i.e., the average number of clauses per sentence (denoted as $\overline{\#C/S}$) and the average number of tokens per clause ($\overline{\#T/C}$). These two statistics of the datasets both characterize the complexity of the clausal structure inside a sentence, where

- $\overline{\#C/S}$ shows the number of complex sentences with more than one clause;
- $\overline{\#T/C}$ depicts the latent dependency distance between two tokens from different clauses.

We also present the performance boosts of our HCA-based parser against *SPRING* in Figure 6. As can be observed, the higher the values of $\overline{\#C/S}$ and $\overline{\#T/C}$ in an OOD dataset, the higher the Smatch improvements that are achieved by our HCA-based approaches. Specifically, New3 and Bio seem to cover more complex texts from newswire and biomedical articles, while TLP contains simpler sentences that are easy for children to read. Therefore, our AMR parser performs much better on complex sentences from Bio and New3, indicating the effectiveness of our HCA-based approaches on LDDs.

5.3. Ablation Study

In the HCA-SA approach, two token visibility matrices derived from HCA trees are introduced to mask certain attention heads. Additionally, we propose a clause-relation-bound attention head setting to integrate inter-clause relations in the encoder. Therefore,

we conduct ablation studies by introducing random token visibility matrices (denoted as “w/o VisMask”) and removing the clause-relation-bound attention setting (denoted as “w/o ClauRel”). Note that “w/o VisMask” contains the case of “w/o ClauRel” because the clause-relation-bound attention setting is based on the masked-self-attention mechanism.

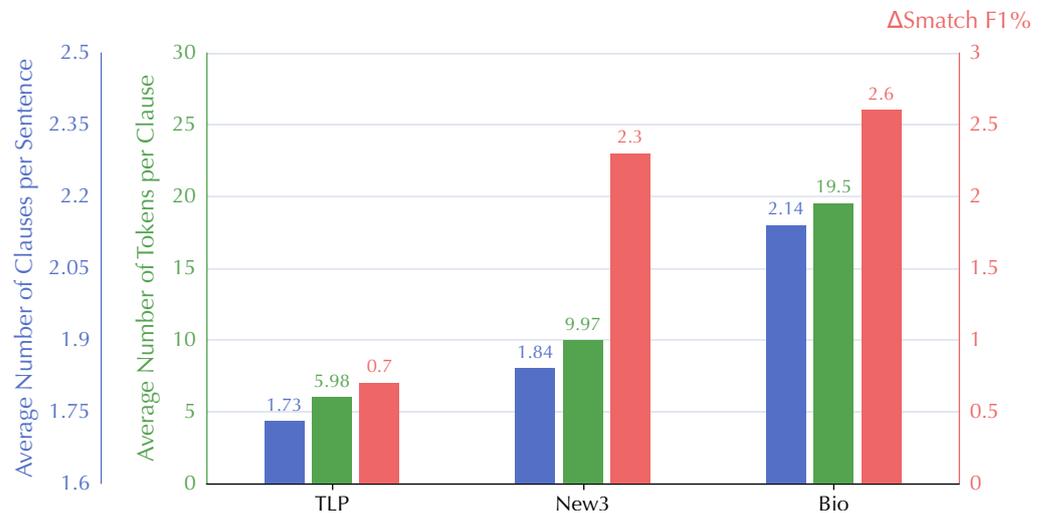


Figure 6. Two important characteristics of three different out-of-distribution (OOD) test sets (i.e., TLP, New3, and Bio) and performance boosts of our HCA-based parser on each test set. The blue and green statistics of each dataset represent the average number of clauses per sentence and tokens per clause, respectively. The red statistics show the improvements of our HCA-based model against the baseline model, *SPRING*, on each OOD dataset.

In the HCA-CL approach, extra training epochs for *Clause-Number* and *Tree-Depth* curricula serve as a warm-up stage for the subsequent training process. To eliminate the effect of the extra epochs, we also add the same number of training epochs to the ablation study of our HCA-CL approach.

As shown in Table 6:

- In HCA-SA, the clause-relation-bound attention setting (denoted as “ClauRel”) contributes most in the *SRL* metric due to the mappings between inter-clause relations (e.g., *Subjective* and *Objective*) and *SRL*-type AMR relations (e.g., :ARG0 and :ARG1).
- In HCA-SA, the masked-self-attention mechanism (denoted as “VisMask”) achieves significant improvements in the *Reent.* metric by increasing the visibility of pronoun tokens to all tokens.
- In HCA-CL, the *Tree-Depth* curriculum (denoted as “TD”) has no effects on the parsing performances. We conjecture that sentences with much deeper clausal structures are rare, and the number of split buckets for the depth of clausal trees is not big enough to distinguish the training sentences.

5.4. Case Study

To further demonstrate the effectiveness of our HCA-based approaches on LDDs in AMR parsing, we compare the output AMR graphs of the same example sentence in Figure 1 parsed by the baseline model *SPRING* and by the modified *SPRING* that applies our HCA-SA and HCA-CL approaches (denoted as *Ours*), respectively, in Figure 7.

For *SPRING*, it mislabels node “go-02” in subgraph G_3 as the :ARG1 role of node “contrast-01”. Then, it fails to realize that it is “anxious” in G_2 that takes the :ARG1 role of “go-02” in G_3 . Additionally, the causality between G_4 and G_5 is not interpreted correctly due to the absence of node “cause-01” and its arguments.

In contrast, when integrating the HCA, *Ours* seems to understand the inter-clause relations better. Although “possible-01” in subgraph G_5 is mislabeled as the :ARG2 role of node “contrast-01”, it succeeds in avoiding the errors made by *SPRING*. Another mistake in

Ours is that the relation :quant between “much” and “anxiety” is reversed and replaced by :domain, which barely impacts the Smatch F1 scores. The vast performance gap between SPRING and our HCA-based SPRING in Smatch F1 scores (66.8% vs. 88.7%) also proves the effectiveness of the HCA on LDDs in AMR parsing.

Table 6. F1 scores (%) of Smatch and three structure-dependent metrics achieved by our HCA-based models in ablation studies on AMR 2.0. “w/o” denotes “without”. “VisMask” and “ClauRel” indicate “token visibility matrices” and “clause-relation-bound attention head setting” in the HCA-based self-attention (HCA-SA) approach. “CN” and “TD” represent the Clause-Number and Tree-Depth curricula in the HCA-based curriculum learning (HCA-CL) approach. Bold figures indicate the best performance achieved by our model with full features. Figures in red represent the most significant performance degradation when removing a specific feature, while those in cyan denote the slightest degradation.

Model	Smatch	Unlab.	Reent.	SRL	
SPRING (2021) [15]	83.8	86.1	70.8	79.6	
Ours	Full	84.5	87.0	72.5	83.2
	w/o VisMask	84.1	86.5	70.9	81.2
	w/o ClauRel	84.4	86.8	72.4	81.2
	w/o CN	84.2	86.7	72.4	83.1
	w/o TD	84.5	87.0	72.5	83.1
	w/o CN,TD	84.2	86.7	72.4	83.1

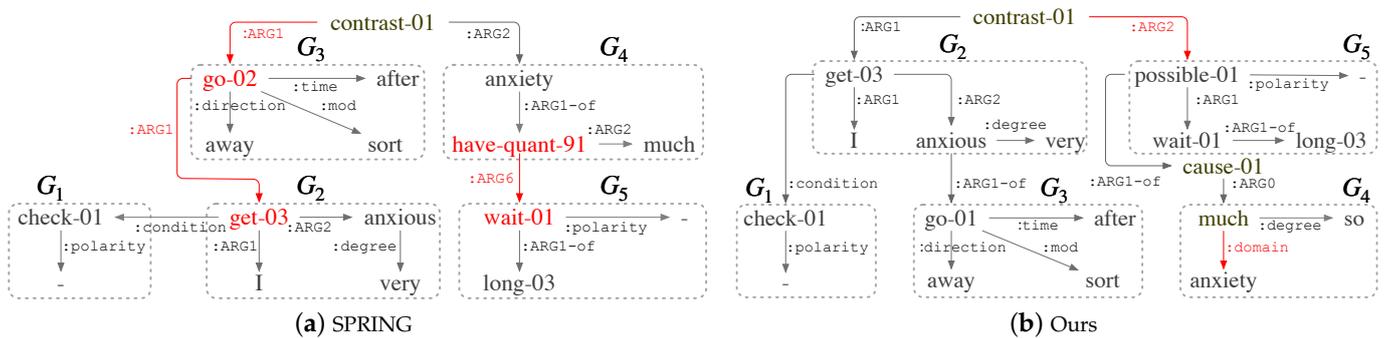


Figure 7. Parsing results of the baseline model SPRING and the modified SPRING that applies our HCA-based approaches (denoted as Ours) when encountering the same AMR 2.0 sentence in Section 1. AMR nodes and edges in red are parsing errors compared to the gold AMR graph. Extra nodes and edges, which are correctly parsed by both, are omitted.

6. Conclusions

We propose two HCA-based approaches, HCA-SA and HCA-CL, to integrate HCA trees of complex sentences for addressing LDDs in AMR parsing. Taking AMR parsing as an example, we apply our HCA-based framework to a popular AMR parser SPRING to integrate HCA features in the encoder. In the evaluations on ID datasets, our parser achieves prominent and explainable improvements against the baseline model, SPRING, and outperforms the SOTA parser, HGAN, in some fine-grained metrics. Notably, as the clause number of the sentence increases, our parser outperforms SPRING by a large margin and achieves better Smatch F1 scores than HGAN, indicating the ability to cure LDDs. In the evaluations of OOD datasets, the performance boosts achieved by our HCA-based approaches are more evident on complicated corpora like New3 and Bio, where sentences consist of more numerous and longer clauses.

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

Funding: This research received no external funding.

Data Availability Statement: The datasets used in our experiments are publicly available: AMR 2.0 at <https://catalog ldc.upenn.edu/LDC2017T10> (accessed on 11 June 2022), AMR 3.0 and its subset New3 at <https://catalog ldc.upenn.edu/LDC2020T02> (accessed on 11 August 2022), TLP at <https://amr.isi.edu/download/amr-bank-struct-v3.0.txt> (accessed on 12 October 2022), Bio at <https://amr.isi.edu/download/2016-03-14/amr-release-test-bio.txt> (accessed on 15 October 2022), and the HCA features of all datasets at <https://github.com/MetroVancloud/HCA-AMRparsing> (accessed on 3 August 2023).

Acknowledgments: Our code extends the GitHub repository *SPRING* at <https://github.com/SapienzaNLP/spring> (accessed on 1 March 2022); thanks to them very much.

Conflicts of Interest: The authors declare that they have no known competing financial interest 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
CL	Curriculum Learning
CPT	Constituency Parse Tree
HCA	Hierarchical Clause Annotation
ID	In-Distribution
IGNN	Implicit Graph Neural Network
LDD	Long-Distance Dependency
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
OOD	Out-of-Distribution
POS	Part-of-Speech
RST	Rhetorical Structure Theory
SA	Self-Attention
SDP	Semantic Dependency
SDPT	Syntactic Dependency Parse Tree
SRL	Semantic Role Labeling
SOTA	State-of-the-Art

References

- Li, Z.; Cai, J.; He, S.; Zhao, H. Seq2seq Dependency Parsing. In Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, NM, USA, 20–26 August 2018; pp. 3203–3214.
- 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, Online, 16–20 November 2020; pp. 1691–1703. [CrossRef]
- He, L.; Lee, K.; Lewis, M.; Zettlemoyer, L. Deep Semantic Role Labeling: What Works and What’s Next. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, BC, Canada, 30 July–4 August 2017; Volume 1, pp. 473–483. [CrossRef]
- Tang, G.; Müller, M.; Rios, A.; Sennrich, R. Why Self-Attention? A Targeted Evaluation of Neural Machine Translation Architectures. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 4263–4272. [CrossRef]
- Jia, Y.; Ye, Y.; Feng, Y.; Lai, Y.; Yan, R.; Zhao, D. Modeling discourse cohesion for discourse parsing via memory network. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018; Volume 2, pp. 438–443. [CrossRef]
- Xu, J.; Gan, Z.; Cheng, Y.; Liu, J. Discourse-Aware Neural Extractive Text Summarization. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 5021–5031. [CrossRef]
- Hihi, S.; Bengio, Y. Hierarchical Recurrent Neural Networks for Long-Term Dependencies. In Proceedings of the Advances in Neural Information Processing Systems, Denver, CO, USA, 27–30 November 1995; MIT Press: Denver, CO, USA, 1995; Volume 8.
- Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]

9. Salton, G.; Ross, R.; Kelleher, J. Attentive Language Models. In Proceedings of the Eighth International Joint Conference on Natural Language Processing, Taipei, Taiwan, 1 December 2017; Volume 1, pp. 441–450.
10. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.u.; Polosukhin, I. Attention is All you Need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Curran Associates, Inc.: Long Beach, CA, USA, 2017; Volume 30.
11. Gu, F.; Chang, H.; Zhu, W.; Sojoudi, S.; El Ghaoui, L. Implicit Graph Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 6–12 December 2020; Curran Associates, Inc.: Vancouver, BC, Canada, 2020; Volume 33, pp. 11984–11995.
12. 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 7th Linguistic Annotation Workshop and Interoperability with Discourse, Sofia, Bulgaria, 8–9 August 2013; pp. 178–186.
13. Peng, X.; Gildea, D.; Satta, G. AMR Parsing with Cache Transition Systems. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32. [[CrossRef](#)]
14. Cai, D.; Lam, W. AMR Parsing via Graph-Sequence Iterative Inference. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 15–20 July 2020; pp. 1290–1301. [[CrossRef](#)]
15. Bevilacqua, M.; Blloshmi, R.; Navigli, R. One SPRING to rule them both: Symmetric AMR semantic parsing and generation without a complex pipeline. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 4–7 February 2021; Volume 35, pp. 12564–12573.
16. Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; Zettlemoyer, L. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 7871–7880. [[CrossRef](#)]
17. Yu, C.; Gildea, D. Sequence-to-sequence AMR Parsing with Ancestor Information. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, Dublin, Ireland, 22–27 May 2022; Volume 2, pp. 571–577. [[CrossRef](#)]
18. Wang, P.; Chen, L.; Liu, T.; Dai, D.; Cao, Y.; Chang, B.; Sui, Z. Hierarchical Curriculum Learning for AMR Parsing. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, Dublin, Ireland, 22–27 May 2022; Volume 2, pp. 333–339. [[CrossRef](#)]
19. Chen, L.; Wang, P.; Xu, R.; Liu, T.; Sui, Z.; Chang, B. ATP: AMRize Then Parse! Enhancing AMR Parsing with PseudoAMRs. In Proceedings of the Findings of the Association for Computational Linguistics: NAACL 2022, Seattle, WA, USA, 10–15 July 2022; pp. 2482–2496. [[CrossRef](#)]
20. 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), Rhodes Island, Greece, 4–9 June 2023; pp. 1–5. [[CrossRef](#)]
21. Carter, R.; McCarthy, M. *Cambridge Grammar of English: A Comprehensive Guide; Spoken and Written English Grammar and Usage*; Cambridge University Press: Cambridge, UK, 2006.
22. Liu, Y.; Ryskin, R.; Futrell, R.; Gibson, E. A verb-frame frequency account of constraints on long-distance dependencies in English. *Cognition* **2022**, *222*, 104902. [[CrossRef](#)] [[PubMed](#)]
23. Mann, W.C.; Thompson, S.A. Rhetorical structure theory: Toward a functional theory of text organization. *Text—Interdiscip. J. Study Discourse* **1988**, *8*, 243–281. [[CrossRef](#)]
24. Fan, Y.; Li, B.; Sataer, Y.; Gao, M.; Shi, C.; Cao, S.; Gao, Z. Hierarchical Clause Annotation: Building a Clause-Level Corpus for Semantic Parsing with Complex Sentences. *Appl. Sci.* **2023**, *13*, 9412. [[CrossRef](#)]
25. Hockett, C.F. A formal statement of morphemic analysis. *Stud. Linguist.* **1952**, *10*, J39.
26. Mahalunkar, A.; Kelleher, J.D. Understanding Recurrent Neural Architectures by Analyzing and Synthesizing Long Distance Dependencies in Benchmark Sequential Datasets. *arXiv* **2020**, arXiv:1810.02966.
27. Xiong, J.; Li, F. Bilevel Topic Model-Based Multitask Learning for Constructed-Responses Multidimensional Automated Scoring and Interpretation. *Educ. Meas. Issues Pract.* **2023**, *42*, 42–61. [[CrossRef](#)]
28. Zukov-Gregoric, A.; Bachrach, Y.; Minkovsky, P.; Coope, S.; Maksak, B. Neural Named Entity Recognition Using a Self-Attention Mechanism. In Proceedings of the 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), Boston, MA, USA, 6–8 November 2017 ; pp. 652–656. [[CrossRef](#)]
29. Li, W.; Qi, F.; Tang, M.; Yu, Z. Bidirectional LSTM with self-attention mechanism and multi-channel features for sentiment classification. *Neurocomputing* **2020**, *387*, 63–77. [[CrossRef](#)]
30. Szubert, I.; Damonte, M.; Cohen, S.B.; Steedman, M. The Role of Reentrancies in Abstract Meaning Representation Parsing. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020, Online, 16–20 November 2020; pp. 2198–2207. [[CrossRef](#)]
31. Flanigan, J.; Thomson, S.; Carbonell, J.; Dyer, C.; Smith, N.A. A Discriminative Graph-Based Parser for the Abstract Meaning Representation. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, Baltimore, MD, USA, 22–27 June 2014; Volume 1, pp. 1426–1436. [[CrossRef](#)]
32. Lyu, C.; Titov, I. AMR Parsing as Graph Prediction with Latent Alignment. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018; Volume 1, pp. 397–407. [[CrossRef](#)]

33. Zhang, S.; Ma, X.; Duh, K.; Van Durme, B. Broad-Coverage Semantic Parsing as Transduction. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; pp. 3786–3798. [[CrossRef](#)]
34. Zhang, S.; Ma, X.; Duh, K.; Van Durme, B. AMR Parsing as Sequence-to-Graph Transduction. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 80–94. [[CrossRef](#)]
35. Konstas, I.; Iyer, S.; Yatskar, M.; Choi, Y.; Zettlemoyer, L. Neural AMR: Sequence-to-Sequence Models for Parsing and Generation. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, BC, Canada, 30 July–4 August 2017; Volume 1, pp. 146–157. [[CrossRef](#)]
36. Peng, X.; Song, L.; Gildea, D.; Satta, G. Sequence-to-sequence Models for Cache Transition Systems. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018; Volume 1, pp. 1842–1852. [[CrossRef](#)]
37. Ge, D.; Li, J.; Zhu, M.; Li, S. Modeling Source Syntax and Semantics for Neural AMR Parsing. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, Macao, China, 10–16 August 2019 Volume 7, pp. 4975–4981. [[CrossRef](#)]
38. Xu, D.; Li, J.; Zhu, M.; Zhang, M.; Zhou, G. Improving AMR Parsing with Sequence-to-Sequence Pre-training. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, 16–20 November 2020; pp. 2501–2511. [[CrossRef](#)]
39. Wang, C.; Xue, N.; Pradhan, S. A Transition-based Algorithm for AMR Parsing. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, CO, USA, 31 May–5 June 2015; pp. 366–375. [[CrossRef](#)]
40. Ballesteros, M.; Al-Onaizan, Y. AMR Parsing using Stack-LSTMs. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 7–11 September 2017; pp. 1269–1275. [[CrossRef](#)]
41. Vilares, D.; Gómez-Rodríguez, C. Transition-based Parsing with Lighter Feed-Forward Networks. In Proceedings of the Second Workshop on Universal Dependencies (UDW 2018), Brussels, Belgium, 1 November 2018; pp. 162–172. [[CrossRef](#)]
42. Naseem, T.; Shah, A.; Wan, H.; Florian, R.; Roukos, S.; Ballesteros, M. Rewarding Smatch: Transition-Based AMR Parsing with Reinforcement Learning. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 4586–4592. [[CrossRef](#)]
43. Fernandez Astudillo, R.; Ballesteros, M.; Naseem, T.; Blodgett, A.; Florian, R. Transition-based Parsing with Stack-Transformers. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020, Online, 16–20 November 2020; pp. 1001–1007. [[CrossRef](#)]
44. Zhou, J.; Naseem, T.; Fernandez Astudillo, R.; Florian, R. AMR Parsing with Action-Pointer Transformer. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 6–11 June 2021; pp. 5585–5598. [[CrossRef](#)]
45. Zhou, J.; Naseem, T.; Fernandez Astudillo, R.; Lee, Y.S.; Florian, R.; Roukos, S. Structure-aware Fine-tuning of Sequence-to-sequence Transformers for Transition-based AMR Parsing. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Online and Punta Cana, Dominican Republic, 7–11 November 2021; pp. 6279–6290.
46. Peng, X.; Song, L.; Gildea, D. A Synchronous Hyperedge Replacement Grammar based approach for AMR parsing. In Proceedings of the Nineteenth Conference on Computational Natural Language Learning, Beijing, China, 30–31 July 2015; pp. 32–41. [[CrossRef](#)]
47. Pust, M.; Hermjakob, U.; Knight, K.; Marcu, D.; May, J. Parsing English into Abstract Meaning Representation Using Syntax-Based Machine Translation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–22 September 2015; pp. 1143–1154. [[CrossRef](#)]
48. Artzi, Y.; Lee, K.; Zettlemoyer, L. Broad-coverage CCG Semantic Parsing with AMR. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–22 September 2015; pp. 1699–1710. [[CrossRef](#)]
49. Groschwitz, J.; Lindemann, M.; Fowlie, M.; Johnson, M.; Koller, A. AMR dependency parsing with a typed semantic algebra. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Melbourne, Australia, 15–20 July 2018; Volume 1, pp. 1831–1841. [[CrossRef](#)]
50. Lindemann, M.; Groschwitz, J.; Koller, A. Compositional Semantic Parsing across Graphbanks. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 4576–4585. [[CrossRef](#)]
51. 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 2nd Shared Task on Discourse Relation Parsing and Treebanking (DISRPT 2021), Punta Cana, Dominican Republic, 11 November 2021; pp. 51–62. [[CrossRef](#)]
52. 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, Abu Dhabi, United Arab Emirates, 7–11 December 2022; pp. 6725–6737.
53. Child, R.; Gray, S.; Radford, A.; Sutskever, I. Generating long sequences with sparse transformers. *arXiv* **2019**, arXiv:1904.10509.
54. Liu, W.; Zhou, P.; Zhao, Z.; Wang, Z.; Ju, Q.; Deng, H.; Wang, P. K-bert: Enabling language representation with knowledge graph. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 2901–2908.

55. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum Learning. In Proceedings of the 26th Annual International Conference on Machine Learning—ICML'09, Montreal, QC, Canada, 14–18 June 2009; pp. 41–48. [[CrossRef](#)]
56. May, J.; Priyadarshi, J. SemEval-2017 Task 9: Abstract Meaning Representation Parsing and Generation. In Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), Vancouver, BC, Canada, 3–4 August 2017; pp. 536–545. [[CrossRef](#)]
57. Cai, S.; Knight, K. Smatch: An Evaluation Metric for Semantic Feature Structures. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, Sofia, Bulgaria, 4–9 August 2013; Volume 2, pp. 748–752.
58. Damonte, M.; Cohen, S.B.; Satta, G. An Incremental Parser for Abstract Meaning Representation. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Valencia, Spain, 3–7 April 2017; Volume 1, pp. 536–546.
59. Riezler, S.; Maxwell, J.T. On Some Pitfalls in Automatic Evaluation and Significance Testing for MT. In Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, Ann Arbor, MI, USA, 25–30 June 2005; pp. 57–64.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.