Holistic Exploration on Universal Decompositional Semantic Parsing: Architecture, Data Augmentation, and LLM Paradigm

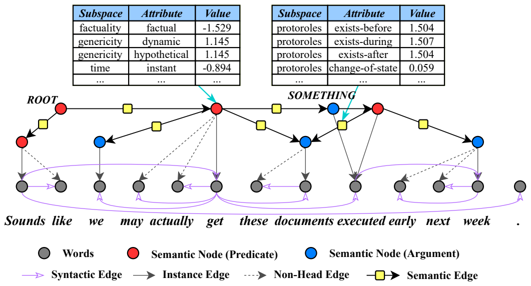
Hexuan Deng, Xin Zhang, Meishan Zhang[*], Xuebo Liu, Min Zhang
Institute of Computing and Intelligence, Harbin Institute of Technology, Shenzhen, China
hxuandeng@gmail.com,zhangxin2023@stu.hit.edu.cn
{zhangmeishan,liuxuebo,zhangmin2021}@hit.edu.cn

Abstract

In this paper, we conduct a holistic exploration of Universal Decompositional Semantic (UDS) parsing, aiming to provide a more efficient and effective solution for semantic parsing and to envision the development prospects after the emergence of large language models (LLMs). To achieve this, we first introduce a cascade model for UDS parsing that decomposes the complex task into semantically appropriate subtasks. Our approach outperforms prior models while significantly reducing inference time. Furthermore, to further exploit the hierarchical and automated annotation process of UDS, we explore the use of syntactic information and pseudo-labels, both of which enhance UDS parsing. Lastly, we investigate ChatGPT's efficacy in handling the UDS task, highlighting its proficiency in attribute parsing but struggles in relation parsing, revealing that small parsing models still hold research significance. Our code is available at https://github.com/hexuandeng/HExp4UDS.

1 Introduction

A long-standing objective in natural language understanding is to create a structured graph of linguistic meaning. Various efforts have been made to encode semantic relations and attributes into a semantic graph, such as Abstract Meaning Representation (AMR; Banarescu et al., 2013), Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport, 2013), and Semantic Dependency Parsing formalisms (SDP; Oepen et al., 2014, 2016). Recently, Universal Decompositional Semantics (UDS; White et al., 2020) introduced a more advanced hierarchical approach, as shown in Figure 1. It can automatically construct semantic relations from syntactic annotations (Zhang et al., 2017) and annotates semantic attributes following decompositional semantics (Reisinger et al., 2015), using only simple questions about words or phrases, significantly lowering the annotation cost.



Subspace	Attribute	Value
factuality	factual	-1.529
genericity	dynamic	1.145
genericity	hypothetical	1.145
time	instant	-0.894
...	...	...

Subspace	Attribute	Value
protoroles	exists-before	1.504
protoroles	exists-during	1.507
protoroles	exists-after	1.504
protoroles	change-of-state	0.059
...	...	...

ROOT
SOMETHING
Sounds like we may actually get these documents executed early next week .
Words
Semantic Node (Predicate)
Semantic Node (Argument)
Syntactic Edge
Instance Edge
Non-Head Edge
Semantic Edge

Figure 1: An example of UDS datasets with syntactic tree and semantic graph. Syntactic tree corresponds to the gray nodes and purple edges, semantic relations correspond to the red and blue nodes as well as the yellow edges, and semantic attributes are in the tables.

However, existing solutions on UDS parsing remain suboptimal. Previous parsing approaches mainly rely on the Seq2Seq transduction framework (Stengel-Eskin et al., 2020), suffering from poor parallelism and long inference times that increase with sentence length. In this paper, we propose a cascade architecture that decomposes the complex parsing task into multiple semantically appropriate subtasks. Within each subtask, our model predicts all corresponding sentence elements simultaneously, enhancing parallelism and substantially reducing inference time. Experimental results show that our approach outperforms previous models while maintaining high efficacy during inference.

Furthermore, the structured data and highly automated annotation procedure of the UDS dataset enable low-cost data augmentation schemes. To take full advantage of this, we design two approaches. Firstly, we incorporate syntactic information, proven beneficial to various tasks (Zaremoodi et al., 2018; Zhang et al., 2020; Stengel-Eskin et al., 2021; Deng et al., 2023). We use multi-task training (Caruana, 1997) to incorporate syntactic information, and propose several approaches for further improvement. Secondly, to utilize the automated

[*]Corresponding author: Meishan Zhang.

Proceedings of the 10th SIGHAN Workshop on Chinese Language Processing (SIGHAN-10), pages 45–57
August 16, 2024 ©2024 Association for Computational Linguistics

tool in UDS that derives semantic relation from syntax, PredPatt (Zhang et al., 2017), Stengel-Eskin et al. (2020) attempted to introduce this tool during inference but do not achieve improvements. In contrast, we propose a data augmentation method that effectively exploits PredPatt's capabilities, leading to significant performance gains in relation parsing. Detailed analysis is also provided to guide the design of parsing systems.

Lastly, we reveal the strengths and weaknesses of large language models (LLMs), such as ChatGPT (Bubeck et al., 2023), in semantic parsing tasks, providing guidance for future developments in this field. LLMs have shown considerable performance in various tasks (Jiao et al., 2023; Wu et al., 2023; Li et al., 2023). For semantic parsing tasks, we conduct preliminary experiments on ChatGPT for the UDS task, directly applying it for parsing, as well as using it for data augmentation. Due to poor performance when directly answering by ChatGPT, we carefully design prompts, breaking down questions into small steps. Results show that ChatGPT excels in attribute parsing but struggles in relation parsing, revealing that small, specialized models still hold research significance.

2 Background and Related Work

UDS Datasets Silveira et al. (2014) create a standard set of Stanford dependency annotations for the English Web Treebank (EWT, Silveira et al., 2014) corpus. Subsequently, White et al. (2016) proposed a framework aimed at constructing and deploying cross-linguistically robust semantic annotation protocols and proposed annotations on top of the EWT corpus using PredPatt (White et al., 2016; Zhang et al., 2017). Several works have then been proposed to provide semantic annotations within this framework, including annotations for semantic roles (Reisinger et al., 2015), entity types (White et al., 2016), event factuality (Rudinger et al., 2018), linguistic expressions of generalizations about entities and events (Govindarajan et al., 2019), and temporal properties of relations between events (Vashishtha et al., 2019). All of these efforts culminated in White et al. (2020), which presents the first unified decompositional semantics-aligned dataset, namely, Universal Decompositional Semantics (UDS).

UDS Parser UDS parsing has been conducted using transition-based parser (Chen and Manning, 2014), deep biaffine attention parser (Dozat and Manning, 2017), and sequence-to-graph transductive parser (Stengel-Eskin et al., 2020). The latter significantly outperforms the others by employing an efficient Seq2Seq transduction framework (Sutskever et al., 2014; Bahdanau et al., 2015). This approach is initially used in AMR parsing (Zhang et al., 2019a) and later extended to cover other semantic frameworks, such as UCCA and SDP, by Zhang et al. (2019b) in a unified transduction framework, which predicts nodes and corresponding edges simultaneously in a Seq2Seq manner. For UDS, an attribute module is added by Stengel-Eskin et al. (2020). Syntactic information is incorporated by Stengel-Eskin et al. (2021), yielding further improvements. Despite these attempts, cascade models with better parallelism and shorter inference time have not yet been explored.

Incorporating Syntactic Information Syntactic information has been shown to improve the performance of downstream tasks. Multi-task learning is widely used to incorporate syntactic information. Hershcovich et al. (2018) improve the performance of semantic parsing by using multi-task learning, with syntactic and other semantic parsing tasks serving as auxiliary tasks. Zaremoodi et al. (2018) use syntactic and semantic information to improve the efficacy of two low-resource translation tasks. Stengel-Eskin et al. (2021) employ a single model to parse syntactic and semantic information simultaneously to improve semantic parsing. Graph convolutional networks (GCN, Kipf and Welling, 2017) are also widely used. Marcheggiani and Titov (2017) use GCNs to incorporate syntactic information in neural models and construct a syntax-aware semantic role labeling model. Zhang et al. (2018) propose an extension of GCNs to help relation extraction models capture long-range relations between words. Zhang et al. (2020) presents a syntax-aware approach based on dependency GCNs to improve opinion role labeling tasks.

3 Preliminaries

The UDS dataset comprises three layers of annotations: syntactic annotations, semantic relation annotations, and decompositional semantic attribute annotations at the edge and node levels.

Syntactic Annotation is derived from the EWT dataset, which provides consistent annotation of grammar, including part-of-speech (*POS*) tag, morphological features, and syntactic dependencies, for human languages. These annotations are used to construct the *syntactic tree*, where each word is tied to a node. As shown in Figure 1, the headword of "we" is "get", and the headword of the root "Sounds" is defined as itself.

Semantic Relations consist of predicates, arguments, and edges between them, which forms the *semantic relations*. It is generated by Predpatt tool (Zhang et al., 2017) automatically, using the POS tag and the syntactic tree as input. Each semantic node explicitly corresponds to one word in the sentence called the center word, demonstrated by the instance edge. Additionally, each semantic node is also tied with several non-repetitive words with the non-head edge, which forms a multi-word span. As shown in Figure 1, the leftmost predict node has a span "Sounds like" with the center word "Sounds". Note that two semantic nodes may correspond to the same word in the case of clausal embedding. Then, an extra argument node "SOMETHING" is introduced as the root of the clause, e.g., "executed" corresponds to an extra argument node.

Semantic Attributes consist of crowdsourced decompositional annotations tied to the semantic relations, detailed in §2. These annotations can be further categorized into node-level and edge-level attributes, corresponding to the table on the left and right in Figure 1, respectively. For each node or edge, all attributes have a value in range $[-3, 3]$. Besides, each attribute also has a confidence in range $[0, 1]$, which shows how likely it is to have the property. Following Stengel-Eskin et al. (2020), we discretized it into $\{0, 1\}$ by setting every non-zero confidence to one.

4 Methodology

In this section, we introduce our cascade model and methods to improve its performance.

4.1 Efficient Cascade Model

As discussed in §3, our goal is to predict syntactic information (POS tags and syntactic tree), semantic relations (semantic nodes, edges, and spans), and semantic attributes (node- and edge-level) using a single model. To this end, we propose a cascade model to predict all of these information step by step, as illustrated in Figure 2. In the following paragraphs, we discuss each component of our model in detail. The sentence is represented as $x_1, x_2, \ldots, x_K$, where x_i represents the i-th word. The properties of the node are represented as t, the properties of the edge as e, the $\mathrm{softmax}$ function as σ, and the ReLU function as R.

Encoder Module embeds each word x_i into a corresponding context-aware representation h_i. We utilize three types of encoders: multi-layer BiLSTM, transformer encoder, and Pre-trained Language Model (BERT, Devlin et al., 2019). For BiLSTM and transformer encoder, we employ a similar embedding layer with Stengel-Eskin et al. (2021) to ensure comparability, concatenating GloVe word embeddings (Pennington et al., 2014), character CNN embeddings, and BERT contextual embeddings. For BERT encoder, we use the default subword embeddings layer and mean-pool over all subwords to obtain the word-level representations.

Syntactic Module predicts the part-of-speech (POS) tag and the syntactic tree. For POS, we use a simple multi-layer perceptron (MLP) over each word representation h_i. For the syntactic tree, each word has exactly one syntactic head, so we predict the headword x_i^y and the corresponding edge type t_i^y for each word x_i. We follow the approach of Dozat and Manning (2017) and Zhang et al. (2019b) to use a biaffine parser, formally:

$$\begin{aligned} \hat{x}_i^y &= p(x|x_i) \\ &= \sigma(\text{Biaffine}(R(w_l^y h_i), R(w_r^y h_{1:K}))) \\ \hat{t}_i^y &= p(t_y|x_i, \hat{x}_i^h) \\ &= \sigma(\text{Bilinear}(R(w_l^t h_i), R(w_r^t \hat{h}_i^y))) \end{aligned} \tag{1}$$

where $x \in \{x_1, x_2, \ldots, x_K\}$, and $\hat{h}_i^y$ is the representation of the predicted head $\hat{x}_i^y$.

Word Classification Module predicts the semantic edge directly connected to each word (instance, non-head) and the type of the parent node. We simplify this edge prediction problem into a classification problem. We define:

- Type "Φ": Words with no connecting edge;
- Type "Syn": Words connect with a non-head edge;
- Type "Pre": Words connect with an instance edge, and its parent is a predicate node;
- Type "Arg": Words connect with an instance edge, and its parent is an argument node;
- Type "Pre + Arg": Words connect with two instance edges, and their parents are a predicate node and an argument node;

We use a simple MLP for classification, formally:

$$\hat{t}_i^m = p(t_m|x_i) = \sigma(\mathrm{MLP}(h_i)) \tag{2}$$

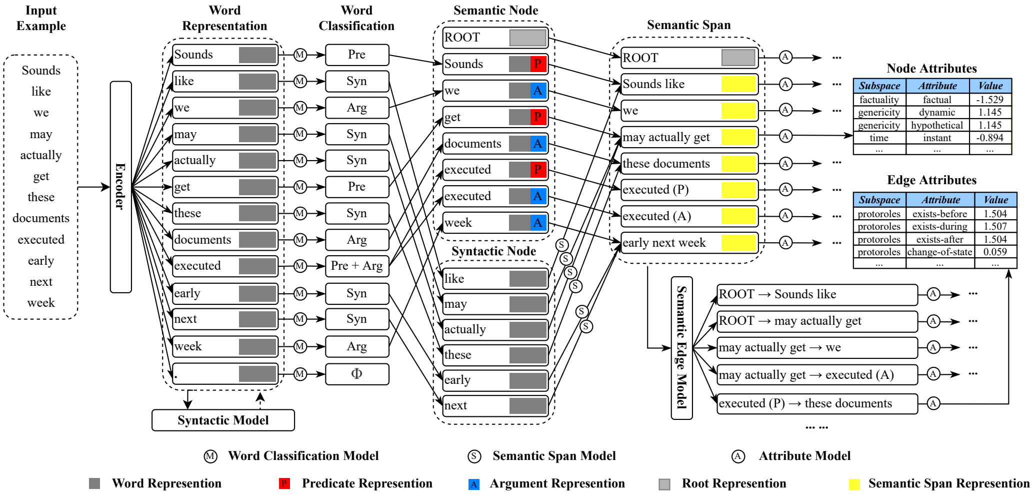


Figure 2: The data flow of our cascade model. The detailed definition of every block is shown in §4.1.

Node Generation predicts the syntactic nodes, semantic nodes, and their corresponding node embeddings. Syntactic nodes $n_1, n_2, \ldots, n_N$ have the label "Syn", and we define the embedding g_i^n of the syntactic node n_i the same as its word embeddings. semantic nodes $m_1, m_2, \ldots, m_M$ have label "Pre", "Arg", or "Pre + Arg". We generate two nodes for "Pre + Arg". So for node embeddings, we first concatenate a node type embedding with its word embedding to distinguish whether it is an argument or predicate node. Then we project it back to the previous dimension with a linear layer to generate the embedding g_i^m of the semantic node m_i. Furthermore, we generate a virtual root node for every sentence with the same trainable embeddings.

Semantic Span Module predicts the semantic span by separating each syntactic node from the semantic nodes. Each syntactic node belongs to exactly one semantic node. So we use the same model as Eq. 1 to predicted which semantic node m_i^h is the syntactic node n_i belongs to, formally:

$$\begin{aligned}\hat{m}_i^h =& p(m|n_i)\\ =&\sigma(\text{Biaffine}(R(w_l^m g_i^n), R(w_r^m g_{1:M}^m)))\end{aligned} \tag{3}$$

where $m \in \{m_1, m_2, \ldots, m_M\}$. The new span level embedding g_i^s for the semantic node m_i is the same as g_i^m by default. Besides, we have also tried to refine g_i^s with the syntactic node embedding, which does not achieve obvious effects.

Semantic Edge Module predicts the edge and the corresponding type $e_{i,j}^m$ between any two semantic nodes. We consider the case where there is no edge between two semantic nodes as a special type Φ. For prediction, we consider the span level embedding for each pair of nodes, formalized as:

$$\begin{aligned}\hat{e}_{i,j}^m =& p(e_m|m_i, m_j)\\ =&\sigma(\text{Biaffine}(R(w_l^e g_i^s), R(w_r^e g_j^s)))\end{aligned} \tag{4}$$

Attribute Module predicts the node-level attributes $\hat{t}_i^a$ for node m_i, and edge-level attributes $\hat{e}_{i,j}^a$ for edge between m_i and m_j. We use the MLP model as the main part, formalized as follows:

$$\begin{aligned}\hat{t}_i^a =& \text{MLP}(g_i^s)\\ v_i =& R(w_l^v g_i^s),\ v_j = R(w_r^v g_j^s)\\ \hat{e}_{i,j}^a =& \text{MLP}([v_i^T W v_j, v_i, v_j])\end{aligned} \tag{5}$$

Here, $W \in \mathbb{R}^{d_v \times d_v \times d_o}$, where d_v is the dimension of v_i and v_j, and d_o is the output dimension. i, j must satisfy $\hat{e}_{i,j}^m \neq \phi$ for $\hat{e}_{i,j}^a$ (edge exists). Note that attributes may not exist, and we use the same model as above to predict the mask of attributes.

Loss To train our models, we use different loss functions depending on the task. For word classification, semantic span, and semantic edge modules, we use cross-entropy loss. For the attribute module, when predicting the mask, we use binary cross-entropy loss. When predicting the attribute, we follow Stengel-Eskin et al. (2020) to use a composite loss function $\mathcal{L}$ for the values, formally:

$$\mathcal{L}_{attr}^{value}\left(\hat{t}, t\right) = \frac{2 \cdot \mathcal{L}_{\text{MSE}}\left(\hat{t}, t\right) \cdot \mathcal{L}_{\text{BCE}}\left(\hat{t}, t\right)}{\mathcal{L}_{\text{MSE}}\left(\hat{t}, t\right) + \mathcal{L}_{\text{BCE}}\left(\hat{t}, t\right)} \tag{6}$$

where $\mathcal{L}_{\text{MSE}}$ is the mean squared loss, $\mathcal{L}_{\text{BCE}}$ is the binary cross-entropy loss, t is the gold attribute, and $\hat{t}$ is our prediction. $\mathcal{L}_{\text{MSE}}$ encourages the predicted attribute value to be close to the true value, while $\mathcal{L}_{\mathrm{BCE}}$ encourages the predicted and reference values to share the same sign.

Finally, to handle this multi-task problem, we use a weighted sum of all the loss functions mentioned above for our model:

$$\begin{aligned}\mathcal{L} =&a_1\mathcal{L}_{cls} + a_2\mathcal{L}_{span} + a_3\mathcal{L}_{edge}+\\ &a_4\mathcal{L}_{attr}^{mask} + a_5\mathcal{L}_{attr}^{value}\end{aligned} \tag{7}$$

where $a_i = 1$ for $i \in [1, .., 5]$, except $a_2 = 2$.

4.2 Incorporating Syntactic Information

By default, we incorporate syntactic information by *multi-task training*. Additionally, we propose *GCN* and *attention* approaches for a more profound incorporation of syntactic information. Specifically, we utilize the syntactic information to update the word embeddings generated by the encoder. The strategies are as follows:

Multi-task Training We add the loss of the syntactic module to term $\mathcal{L}$, which incorporates syntactic information into the shared encoder through back-propagation. We use cross-entropy loss for POS and syntactic tree parsing, formally:

$$\mathcal{L}_{syn} = \mathcal{L} + a_6\mathcal{L}_{pos} + a_7\mathcal{L}_{tree} \tag{8}$$

where $a_6 = a_7 = 1$ during our experiments.

GCN Inspired by the idea of GCN (Kipf and Welling, 2017), we try to encode the predicted adjacency matrix information into the embedding. In the syntactic tree, we consider two types of edges: directed edges from parent nodes (top) to child nodes (bottom), and those with reverse directions. Then we employ a bidirectional GCN consisting of 1) top-down GCN to convey sentence-level information to local words, and 2) bottom-up GCN to convey phrase-level information to the center word. Additionally, to further convey the edge type information corresponding to the current word, we 3) consider the probability distribution of its edge type, and use a GCN-like method to convey this information. With the word embedding matrix $\mathbf{H}$ being the input $\mathbf{H}^{(0)}$, we use a l layer model (with $l = 2$ in practice), formally:

$$\begin{aligned}\mathbf{V}^{(i)} =&[\mathbf{A}_h\mathbf{H}^{(i)}\mathbf{W}_1^{(i)}, \mathbf{A}_h^T\mathbf{H}^{(i)}\mathbf{W}_2^{(i)}, \mathbf{A}_t\mathbf{T}_e\mathbf{W}_3^{(i)}]\\ \mathbf{H}^{(i+1)} &= R(\mathbf{W}_4^{(i)}R(\mathbf{V}^{(i)}))\\ \mathbf{H}_o &= \mathbf{W}_o[\mathbf{H}^{(0)}, \mathbf{H}^{(l)}]\end{aligned} \tag{9}$$

where $\mathbf{A}_h$ is the top-down adjency matrix prediction, $\mathbf{A}_h^T$ is the bottom-up ones, $\mathbf{A}_t$ is the edge type probability distribution, and $\mathbf{T}_e$ is the trainable edge type embedding matrix. Note that the adjacency matrix does not self-loop, so GCN does not convey information about the words themselves. We then combine the original word embeddings $\mathbf{H}^{(0)}$ with the output $\mathbf{H}^{(l)}$ to get the new word embeddings $\mathbf{H}_o$. Under such a design, good results can be achieved with a relatively shallow network.

Attention Word representation after dimension reduction used in syntactic edge and type prediction contains basic information of the syntactic tree (Stengel-Eskin et al., 2021). So we directly use the representations in Eq. 1, which are used in the Biaffine and Bilinear model. Formally:

$$\begin{aligned}\mathbf{V} =&R([w_l^w\mathbf{H}, w_r^w\mathbf{H}, w_l^t\mathbf{H}, w_r^t\mathbf{H}])\\ \mathbf{H}_o &= \mathbf{W}_o[\mathbf{H}, \mathbf{A}_h\mathbf{V}]\end{aligned} \tag{10}$$

where all the $w_*^*\mathbf{H}$ come from Eq. 1 without recalculation. Compared to the GCN approach, this method uses fewer new parameters and requires less additional calculation, while still preserving the performance improvements achieved by the GCN approach to some extent.

4.3 Data Augmentation with PredPatt

One of the features of the UDS dataset is the strong correlation with external tools PredPatt. Stengel-Eskin et al. (2020) attempt to use an external model to predict the POS tags and syntactic tree on the test dataset, which are then fed directly to PredPatt to obtain the semantic relations. However, the effectiveness of this method is relatively poor, likely due to two issues: 1) the error transmission problem comes from the prediction of the syntactic model, and 2) the rule-based tools are not as robust as neural networks towards noisy inputs.

To address these issues, we propose a data augmentation method. Instead of using it during inference, we use it to augment the data, with only the help of external unlabeled data. Specifically, we first train a model to predict the syntactic tree and POS tag, using the above syntactic model. Next, we use PredPatt to generate pseudo labels (i.e., semantic relations) for the unlabeled data. Finally, we use these data to pre-train our model, and then fine-tune it with a smaller learning rate using the labeled UDS dataset, which achieves significant improvements in relation parsing.

	Strategy	S-P	S-R	S-F1	Attr. ρ	Attr. F1	UAS	LAS	POS
Baseline	LSTM	89.90	85.85	87.83	0.46	60.41	-	-	-
	+ SYN	88.58	87.67	88.12	0.46	61.28	91.44	88.80	-
	TFMR	90.04	87.98	89.19	0.56	67.89	-	-	-
	+ SYN	**91.09**	89.01	90.04	0.56	66.85	92.40	89.96	-
Ours	LSTM	87.75	91.12†	89.79†	0.47†	57.93	-	-	-
	+ SYN	88.82†	92.50†	90.62†	0.46	57.34	91.71	89.10	96.29
	+ SYN + DA	90.00	93.37	91.65	0.33	49.66	92.65	90.51	96.67
	TFMR	88.34	92.90†	90.56†	0.49	59.68	-	-	-
	+ SYN	89.28	93.56†	91.37†	0.49	58.45	92.07	89.65	96.85
	+ SYN + DA	90.15	93.49	91.79	0.42	54.27	**93.03**	90.91	97.10
	BERT	88.90	92.77†	90.79†	**0.60†**	67.02	-	-	-
	+ SYN	89.51	94.18†	91.79†	0.59†	65.78	92.81†	90.73†	**97.18**
	+ SYN + DA	90.27	**94.23**	**92.20**	0.54	63.91	92.98	**90.93**	97.08
LLM	PRED	35.50	51.28	41.96	-	-	-	-	-
	CASC	38.13	53.26	44.44	-	-	-	-	-
	ATTR	-	-	-	-	**80.69**	-	-	-

Table 1: Main results. "LSTM", "TFMR"(Transformer), "BERT" stands for different encoder. We run t-test against the corresponding baseline, and † means significantly higher with $> 95\%$ confidence. "+SYN" means GCN approach in §4.2, and "+DA" means the data augmentation method in §4.3. Other abbreviations are detailed in §5.1.

5 Experiment and Analysis

5.1 Experimental Setup

Datasets We conduct experiments on the UDS dataset (White et al., 2020), with 10k valid training sentences. For English monolingual data, we use publicly available News Crawl 2021 corpus (Zhang and Zong, 2016; Wu et al., 2019). In the experiment of the data augmentation method, we first generate the pseudo-targets for all the monolingual data, then filter out the ones that have invalid syntactic and semantic graphs. Finally, we randomly select a 100k corpus subset.

LLMs We explore the direct use of LLMs for parsing tasks. First, for semantic relation parsing, we try two types of prompts: 1) The Predpatt prompt guides the LLM to first generate the syntactic parsing, then follow the instructions of the PredPatt tool step by step to generate the semantic relations (PRED). 2) The cascade approach follows the idea of our model to decompose the UDS parsing (CASC), which first selects the center phrase of the semantic node, and then expands every phrase into a span. To make sure that the center word has only one word, we select it at the last step. Second, for semantic attribute parsing, we provide the sentence and the corresponding node/edge as input, definitions of attribute types as instruction, and conduct experiments under the oracle setting defined in §A.2 (ATTR). As the scale of attribute scoring may vary across conversation rounds, we only let it predict positive or negative.

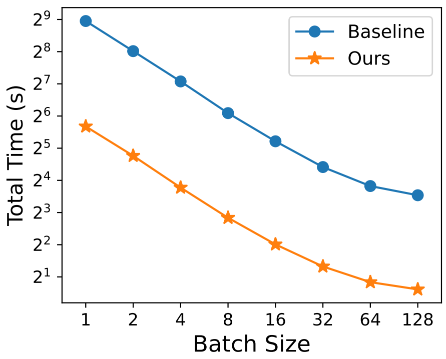


Figure 3: Total inference time for forward propagation of the two models, varying with the batch size. We use logarithmic coordinates for better comparison.

Metrics We follow the setting given by Stengel-Eskin et al. (2021), using S-score for semantic relation prediction, Attribute ρ and F1 for UDS attributes, UAS and LAS for heads and edge types, and POS for part-of-speech.

We use Stengel-Eskin et al. (2021) as the baseline. Details regarding the baseline for comparison, prompts and settings used for ChatGPT, the introduction of metrics, and model training configurations are provided in Appendix A.

5.2 Main Results

We conduct experiments on three types of encoders, as demonstrated in Table 1.

Our cascade model outperforms the baseline model. Under basic settings, our best setting (BERT) significantly improves the baseline (TFMR) in S-F1 (+1.60) and Attr. ρ (+0.04), and

	Strategy	S-P	S-R	S-F1	Attr. ρ	Attr. F1	UAS	LAS	POS
LSTM	Naive	87.75	91.12	89.79	0.47	57.93	-	-	-
	+ Joint	88.21	92.51	90.31	0.45	55.42	91.51	89.07	96.23
	+ Attn.	88.62	92.55	90.54	0.45	57.65	91.95	89.41	96.26
	+ GCN	88.82	92.50	90.62	0.46	57.34	91.71	89.10	96.29
	+ Span	88.45	92.31	90.34	0.47	57.85	91.58	89.08	96.37
TFMR	Naive	88.34	92.90	90.56	0.49	59.68	-	-	-
	+ Joint	88.64	93.53	91.02	0.51	59.56	91.99	89.42	96.60
	+ Attn.	88.82	93.46	91.08	0.49	58.86	91.84	89.35	96.77
	+ GCN	89.28	93.56	91.37	0.49	58.45	92.07	89.65	96.85
	+ Span	88.85	93.19	90.97	0.50	59.46	91.60	89.29	96.71
BERT	Naive	88.90	92.77	90.79	0.60	67.02	-	-	-
	+ Joint	88.87	93.75	91.25	0.60	67.63	92.95	90.79	97.12
	+ Attn.	89.25	94.05	91.59	0.58	66.60	92.94	90.77	97.02
	+ GCN	89.51	94.18	91.79	0.59	65.78	92.81	90.73	97.18
	+ Span	88.95	93.64	91.23	0.59	65.97	92.92	90.74	97.23

Table 2: The effect of different strategies to incorporate syntactic information. "Naive" means no additional syntactic information. "+Joint", "+Attn", and "+GCN" mean incorporating syntactic information using joint training, GCN, and attention in §4.2, separately. "+Span" means refine span embeddings using syntactic nodes.

slightly worse in Attr. F1. The above results are also preserved under +SYN settings (+1.75 and +0.03, respectively). Furthermore, we calculated the total inference time for forward propagation of the two models, averaging on validation and test datasets (about 1.3k sentences). The results are shown in Figure 3 under logarithmic coordinates. Our model significantly reduces the inference time for all batch sizes (9.56 times faster on average). Finally, using additional data augmentation methods, the S-F1 can be further improved (+2.16), which is also held in LSTM and Transformer (+3.53 and +1.75, respectively). The above results show that our model significantly outperforms the baseline.

Syntactic information and data augmentation methods enhance semantic relation parsing. Our model primarily focuses on improving semantic relation parsing, which LLMs are not good at. We summarize the corresponding result in Figure 4. We can see that both the two approaches can significantly improve relation parsing, with +0.88 for syntactic information and +0.62 for the data augmentation method on average. Besides, the improvements are orthogonal to each other and can be used simultaneously, pushing the results of different models towards a similar limit, since lower-performing models experience greater improvements.

The same methods do not benefit attribute prediction. However, our proposed methods for further improvements do not consistently improve the attribute parsing. Attributes derive from crowd-sourced annotation, which is not closely related to the syntactic or semantic information. Thus, syntactic information cannot provide useful information for attribute prediction, and using more data to pretrain a better model for semantic relation parsing is harmful to the performance of attribute parsing.

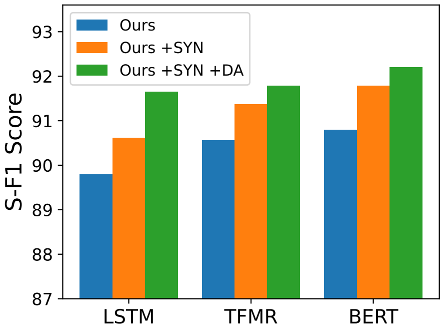


Figure 4: S-F1 score of different encoders. Abbreviations are defined in Table 1.

ChatGPT performs poorly on relation parsing but well on attribute parsing. The generated relations of ChatGPT are typically semantically compliant. However, they struggle to follow instructions step by step, resulting in poor performance on relation parsing. Additionally, data augmentation does not work well for the UDS task with ChatGPT, revealing significant distribution shifts for the data generated by ChatGPT. Despite these shortcomings, its high performance on attribute parsing demonstrates promising syntactic understanding. Detailed analysis can be found in Appendix B.

5.3 Exploration on Syntactic Information

We conduct experiments on different ways to join syntactic information into the model, and the re-

Model	In domain	Predpatt	S-P	S-R	S-F1	Δ	UAS	LAS	POS
Ours	×	×	89.34	94.05	91.63	+0.38	93.05	91.01	97.10
Ours	×	✓	90.15	94.28	92.17	+0.92	93.26	91.28	97.17
Ours	✓	×	89.08	94.03	91.49	+0.24	92.62	90.52	97.19
Ours	✓	✓	89.56	94.37	91.90	+0.65	92.88	90.91	97.22
Stanza	×	✓	89.24	94.12	91.62	+0.37	92.96	90.86	97.27
Stanza	✓	✓	89.69	94.24	91.90	+0.65	92.76	90.74	97.10
ChatGPT	×	✓	88.25	93.28	90.70	-0.55	92.38	90.28	96.99
Syntactic Teacher			-	-	-		93.24	91.14	97.54
Semantic Relation Teacher			88.87	93.75	91.25	-	92.95	90.79	97.12

Table 3: The effect of different data augmentation approaches. "Model" means which teacher model to use, "In domain" means whether to select data with closer domain, and "Predpatt" means whether to use an external tool or simply use the distillation method. "Syntactic Teacher" is trained only on syntactic targets, while "Semantic Relation Teacher" on syntactic and semantic relation targets. Both only use multi-task learning methods.

sults are shown in Table 2.

Syntactic information enhances semantic relation parsing. Our experiments show consistent improvements in S-F1 scores across different methods of integrating syntactic information, with +0.48 for SYN, +0.69 for Contact, and +0.88 for GCN, which is also used as our default settings. However, because of the different syntactic foundations arising from different annotation methods, we do not observe a consistent trend of attribute parsing, aligned with findings in Stengel-Eskin et al. (2021).

Incorporating child syntactic information has less impact on the results. We tried to use a better span representation, which uses a self-attention over all words in the span, instead of using only the representation of the center word. However, the attribute prediction does not achieve consistent improvements. This shows that the center word can well represent the semantics of the whole span, and is the default setting in our experiments.

5.4 Exploration on Data Augmentation

We conduct experiments using the data augmentation method under the basic multi-task training method to incorporate syntactic information, and the results are shown in Table 3.

Data augmentation significantly improves the semantic parsing. Under different ways to incorporate syntactic information, the S-F1 consistently improves, with +0.54 on average and +0.92 for best settings (ours without in-domain and with PredPatt), which is used as the default data augmentation method. Besides, our proposed ways to better utilize the external tool also significantly outperform the basic distillation settings, i.e., +0.48 on average, which shows the efficacy of our methods.

How does the in-domain unlabeled data act? We are also curious about how the domain of the datasets influences the results. We follow the idea of Moore and Lewis (2010) to score the unlabeled data by the difference between the score of the in-domain language model and the language model trained from which the unlabeled data is drawn. We refer the reader to the original paper for further details. Results have shown that for our larger models with better generalization, the in-domain data hurt the performance (-0.27). For the smaller model given in Stanza, the in-domain data performs better (+0.28), while both are worse than the results with our models. This shows that the performance of the teacher model is important, and for models with good generalization, always using in-domain data is not a good choice.

6 Conclusion

In this paper, we conduct a holistic exploration of semantic parsing, focusing on Universal Decompositional Semantic (UDS) parsing. First, we develop an efficient cascade model that offers improved performance and reduced training and inference costs. Additionally, we examine data augmentation methods that incorporate syntactic information and employ the PredPatt tool to strengthen the model's syntactic and semantic comprehension. Lastly, we find that ChatGPT performs poorly in relation parsing and data augmentation but excels in attribute parsing. This reveals that small, specialized models still hold research significance in semantic parsing.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 62176180).

Limitations

This study has several limitations. Firstly, due to computational constraints, the research is confined to using small models for solving semantic parsing problems, resulting in a lack of exploration in fine-tuning with LLMs. Better performance may be achieved with LLMs, given their stronger semantic understanding capabilities. Secondly, although the approach proposed in this study can be applied to various semantic parsing tasks, time constraints led to the selection of only one representative dataset for testing. This restricts a more comprehensive analysis of the proposed approach and LLMs' performance. Furthermore, due to cost constraints and regional lockouts, we were unable to include more LLMs, such as GPT-4, GPT-4o, and Claude, in our analysis. Lastly, before the era of LLMs, semantic parsing was able to enhance the performance of various downstream tasks. However, for LLMs, whether fine-tuning LLMs with semantic parsing datasets or providing semantic trees in the context can improve downstream tasks remains to be explored in future work.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *ACL*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *LAW@ACL*.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: early experiments with gpt-4. *ArXiv*.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *ACL*.

Rich Caruana. 1997. Multitask Learning. *Mach. Learn.*, 28(1):41–75.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*.

Hexuan Deng, Liang Ding, Xuebo Liu, Meishan Zhang, Dacheng Tao, and Min Zhang. 2023. Improving simultaneous machine translation with monolingual data. *AAAI*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *ICLR*.

Venkata Subrahmanyan Govindarajan, Benjamin Van Durme, and Aaron Steven White. 2019. Decomposing generalization: models of generic, habitual and episodic statements. *Trans. Assoc. Comput. Linguistics*.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2018. Multitask parsing across semantic representations. In *ACL*.

Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Xing Wang, and Zhaopeng Tu. 2023. Is chatgpt a good translator? a preliminary study. *ArXiv*.

Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR (Poster)*.

Bo Li, Gexiang Fang, Yang Yang, Quansen Wang, Wei Ye, Wen Zhao, and Shikun Zhang. 2023. Evaluating chatgpt's information extraction capabilities: an assessment of performance, explainability, calibration, and faithfulness. *ArXiv*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *EMNLP*.

Robert C. Moore and William D. Lewis. 2010. Intelligent selection of language model training data. In *ACL*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Zdenka Uresová. 2016. Towards comparability of linguistic graph banks for semantic parsing. In *LREC*.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Dan Flickinger, Jan Hajic, Angelina Ivanova, and Yi Zhang. 2014. Semeval 2014 task 8: broad-coverage semantic dependency parsing. In *SemEval@COLING*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: global vectors for word representation. In *EMNLP*.

Barbara Plank, Héctor Martínez Alonso, Zeljko Agic, Danijela Merkler, and Anders Søgaard. 2015. Do dependency parsing metrics correlate with human judgments? In *CoNLL*.

Dee Ann Reisinger, Rachel Rudinger, Francis Ferraro, Craig Harman, Kyle Rawlins, and Benjamin Van Durme. 2015. Semantic proto-roles. *Trans. Assoc. Comput. Linguistics*.

Rachel Rudinger, Aaron Steven White, and Benjamin Van Durme. 2018. Neural models of factuality. In *NAACL-HLT*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: summarization with pointer-generator networks. In *ACL*.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel R. Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for english. In *LREC*.

Elias Stengel-Eskin, Kenton W. Murray, Sheng Zhang, Aaron Steven White, and Benjamin Van Durme. 2021. Joint universal syntactic and semantic parsing. *Trans. Assoc. Comput. Linguistics*.

Elias Stengel-Eskin, Aaron Steven White, Sheng Zhang, and Benjamin Van Durme. 2020. Universal decompositional semantic parsing. In *ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Siddharth Vashishtha, Benjamin Van Durme, and Aaron Steven White. 2019. Fine-grained temporal relation extraction. In *ACL*.

Aaron Steven White, Dee Ann Reisinger, Keisuke Sakaguchi, Tim Vieira, Sheng Zhang, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2016. Universal decompositional semantics on universal dependencies. In *EMNLP*.

Aaron Steven White, Elias Stengel-Eskin, Siddharth Vashishtha, Venkata Subrahmanyan Govindarajan, Dee Ann Reisinger, Tim Vieira, Keisuke Sakaguchi, Sheng Zhang, Francis Ferraro, Rachel Rudinger, Kyle Rawlins, and Benjamin Van Durme. 2020. The universal decompositional semantics dataset and decomp toolkit. In *LREC*.

Haoran Wu, Wenxuan Wang, Yuxuan Wan, Wenxiang Jiao, and Michael Lyu. 2023. Chatgpt or grammarly? evaluating chatgpt on grammatical error correction benchmark. *ArXiv*.

Lijun Wu, Yiren Wang, Yingce Xia, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2019. Exploiting monolingual data at scale for neural machine translation. In *EMNLP/IJCNLP*.

Poorya Zaremoodi, Wray L. Buntine, and Gholamreza Haffari. 2018. Adaptive knowledge sharing in multi-task learning: improving low-resource neural machine translation. In *ACL*.

Bo Zhang, Yue Zhang, Rui Wang, Zhenghua Li, and Min Zhang. 2020. Syntax-aware opinion role labeling with dependency graph convolutional networks. In *ACL*.

Jiajun Zhang and Chengqing Zong. 2016. Exploiting source-side monolingual data in neural machine translation. In *EMNLP*.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019a. Amr parsing as sequence-to-graph transduction. In *ACL*.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019b. Broad-coverage semantic parsing as transduction. In *EMNLP/IJCNLP*.

Sheng Zhang, Rachel Rudinger, and Benjamin Van Durme. 2017. An evaluation of predpatt and open ie via stage 1 semantic role labeling. In *IWCS*.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *EMNLP*.

A Supplementary Experimental Setup

A.1 Experimental Configurations

Model Training Our model is trained on one NVIDIA A30 Tensor Core GPU with a batch size of 16 and a dropout rate of 0.3. We fix BERT parameters for LSTM and transformer encoders and keep them trainable when BERT itself is the encoder. For the majority of the training process, we set the learning rate to 2e-4, while for BERT encoder, we set it to 1e-5. For a fair comparison, we use a linear projection of the output of all the encoders to unify the output dimension to 1024. We run each model five times under different seeds in the main table and show the average score.

Baseline We use Stengel-Eskin et al. (2021) as the baseline. It first employs GloVe word embeddings (Pennington et al., 2014), character CNN embeddings, and BERT (Devlin et al., 2019) to generate the context-aware representations of the input sentence. Then, it generates each edge with a decoder in an autoregressive way, following the idea of a pointer-generator network (See et al., 2017). After that, it uses a deep biaffine (Dozat and Manning, 2017) graph-based parser to create edges. Node- and edge-level attributes are then predicted after every step, with a multi-layer perception for node attributes and a deep biaffine for edge attributes. Besides, the introduction of syntactic information is preliminarily tried, and we only report their optimal results for each metric.

LLMs For ChatGPT, we conduct experiments in the dialog box, using the ChatGPT Mar 23 Version in 2023. We provide details on the prompts used for evaluating ChatGPT under UDS tasks in Figure 5.

A.2 Metrics

We follow the setting given by Stengel-Eskin et al. (2021), detailed as follows.

S-score This metric measures performance on the semantic relation prediction task. Following the Smatch metric (Cai and Knight, 2013), which uses a hill-climbing approach to find an approximate graph matching between a reference and predicted graph, S-score (Zhang et al., 2017) provides precision (S-P), recall (S-R), and F1 score (S-F1) for nodes, edges, and attributes. We follow Stengel-Eskin et al. (2021) and evaluate the S-score for nodes and edges only, which evaluates against full UDS arborescences with linearized syntactic subtrees included as children of semantic heads.

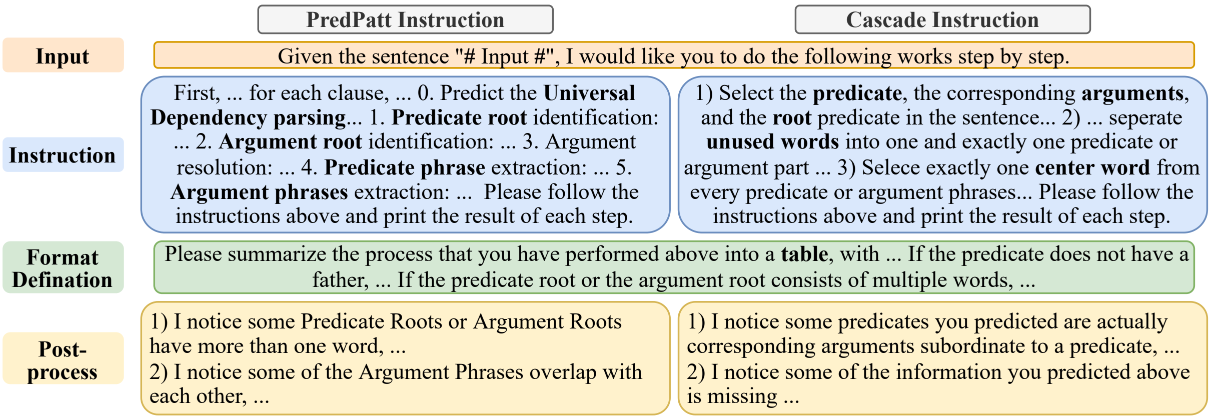
Attribute ρ & F1 For UDS attributes, we use the pearson correlation ρ (Attr. ρ) between the predicted attributes at each node and the gold annotations in the UDS corpus. We also use F1-score (Attr. F1) to measure whether the direction of the attributes matches that of the gold annotations. We binarized the attribute with threshold value $\theta = 0$ for gold attributes, and tune θ for predicted ones per attribute type on validation data. Both of them are obtained under an "oracle" setting, where the gold graph structure is provided.

Syntactic Metric We follow Plank et al. (2015) to use Unlabeled Attachment Score (UAS) to compute the fraction of words with correctly assigned heads, and Labeled Attachment Score (LAS) to compute the fraction with correct heads and edge types. While for part-of-speech (POS), we simply use the accuracy of prediction.

B Exploration on LLM Paradigm

ChatGPT performs poorly on relation parsing. For semantic relation parsing, we use the prompt given in §A.1 3 times, which generates 9 different results. We filter out invalid output (no table or table with incorrect headers) and select the best result for each sentence. There are still 11.04% and 0.37% of the sentences that do not have correct results for PRED and CASC, respectively, which are filtered out. Despite this favorable setting, it still achieved poor results. Under our observation, the generated relations of LLMs are typically semantically compliant. However, they struggle to follow the instructions step by step, leading to outputs that often do not meet our requirements, and repetitions and incorrect summarizations in the table also commonly occur. As a result, LLMs perform poorly on relation parsing, especially in precision, and complex post-processing constructed by professionals is highly required.

ChatGPT performs perfectly on attribute parsing. For semantic attribute parsing, we only run ChatGPT once. 3.03% of the sentences do not have correct results and are filtered out. Results show that ChatGPT significantly outperforms the small models, achieving a +12.80 increase in Attribute F1 scores compared to the best model. We think that for ChatGPT, which is well-aligned with humans, it is easier to predict the attributes given by



Given the sentence "# Input #", I would like you to do the following works step by step.

First, ... for each clause, ... 0. Predict the **Universal Dependency parsing**... 1. **Predicate root** identification: ... 2. **Argument root** identification: ... 3. Argument resolution: ... 4. **Predicate phrase** extraction: ... 5. **Argument phrases** extraction: ... Please follow the instructions above and print the result of each step.

1) Select the **predicate**, the corresponding **arguments**, and the **root** predicate in the sentence... 2) ... seperate **unused words** into one and exactly one predicate or argument part ... 3) Selece exactly one **center word** from every predicate or argument phrases... Please follow the instructions above and print the result of each step.

Please summarize the process that you have performed above into a **table**, with ... If the predicate does not have a father, ... If the predicate root or the argument root consists of multiple words, ...

1) I notice some Predicate Roots or Argument Roots have more than one word, ...
2) I notice some of the Argument Phrases overlap with each other, ...

1) I notice some predicates you predicted are actually corresponding arguments subordinate to a predicate, ...
2) I notice some of the information you predicted above is missing ...

Figure 5: The prompt for semantic relation parsing for ChatGPT. For each generation, we first input the input and instruction, then input the format definition to get the prediction. Finally, we input the post-process part one by one to generate better predictions, i.e., three predictions for a sentence in a single conversation.

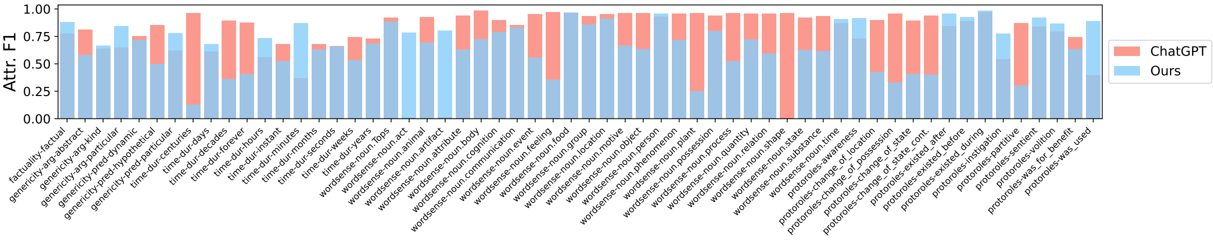


Figure 6: Attribute F1 score for each UDS attribute using ChatGPT (80.69 on average) or using our basic BERT model (67.02 on average). The x-axis is the UDS attribute name, with the ones beginning with "protoroles" being the edge-level attributes (the rightmost 14 attributes), and other attributes are at the node level.

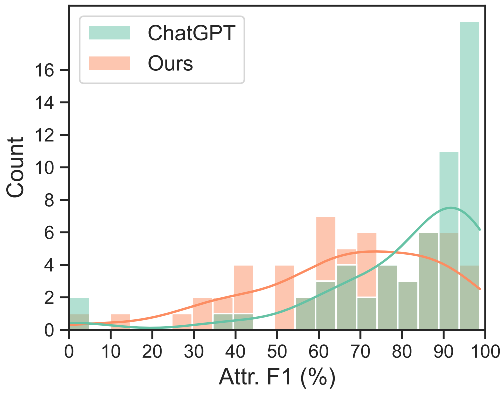


Figure 7: The Attribute F1 score distribution across each UDS attribute. "ChatGPT" stands for prompted with ATTR settings, and "Ours" represents our best strategy under attribute prediction.

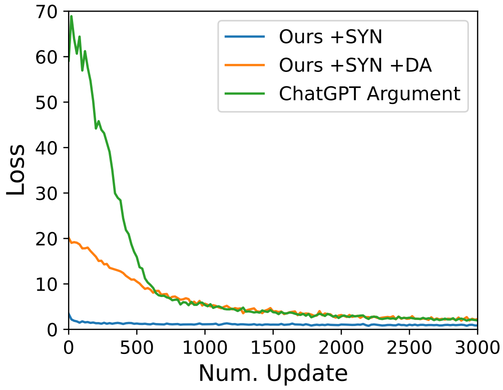


Figure 8: Loss function curve over the first 3k steps. The defination of "+SYN" and "+DA" are align with ones in Table 1.

human annotators rather than the long logical chain reasoning task. In addition, only need to predict positive and negative without considering the pearson correlation is also one of its advantages.

For further verification, we calculate the Attribute F1 scores for all attributes in Figure 6. We can observe that ChatGPT performs well on most of the attributes when compared to our model, with 60.34% and 25.86% of the attributes respectively having F1 scores above 85%. Furthermore, ChatGPT performs perfectly on word-sense attributes, achieving an F1 score of 86.99. In contrast, our models do not display significantly superior results, with an F1 score of 68.49. For a clearer comparison, we have compiled the distribution of Attribute F1 values for different attributes for our best-trained small model and ChatGPT, as shown in Figure 7.

We can see that ChatGPT scores higher on a greater number of attributes. We believe that with more detailed guidance and rigorous post-processing, LLMs have the potential to replace humans in annotation tasks.

How does data augmentation with ChatGPT act? We investigate the generation of new data for downstream model training, which is widely used. We use the random token lists as input rather than the unlabeled data, and let LLM generate the POS tag and syntax tree, which are further used to generate pseudo-labels, following §4.3. Since Universal Dependencies is a widely used dataset and contains both of the required information, a simple prompt can be used. For ChatGPT generation, we select a 10k corpus subset.

LLMs do not perform well in semantic relation generation, and directly using external tools to assist the test set is not effective (Stengel-Eskin et al., 2020), so it is natural to think of using LLM to augment the data in a similar way. We used the zero-shot settings, detailed in §A.1. However, the performance has declined. For further analysis, we propose the training loss for the first 3k updates for different models in Figure 8. We can see that our data augmentation method can significantly lower the initial training loss, which shows that similar data distribution is shared between our proposed pseudo-labeled data and the training data. However, the initial loss of ChatGPT argumentation is even higher than random initializing (Ours +SYN). This shows significant distribution shifts for the data generated by ChatGPT, which shows the need for more detailed prompts and ways to select properly generated data.