Few-Shot Semantic Dependency Parsing via Graph Contrastive Learning

Bin Li[1,2], **Yunlong Fan**[1,2], **Yikemaiti Sataer**[1,2], **Chuanqi Shi** [1,2], **Miao Gao** [1,2], **Zhiqiang Gao**[1,2,*]
[1]School of Computer Science and Engineering, Southeast University, Nanjing 210096, China
[2]Key Laboratory of Computer Network and Information Integration, Ministry of Education, China
{lib, fanyunlong, yikmat, chuanqi_shi, miaogao, zqgao}@seu.edu.cn

Abstract

Graph neural networks (GNNs) have achieved promising performance on semantic dependency parsing (SDP), owing to their powerful graph representation learning ability. However, training a high-performing GNN-based model requires a large amount of labeled data and it is prone to over-fitting in the absence of sufficient labeled data. To address this drawback, we propose a syntax-guided graph contrastive learning framework to pre-train GNNs with plenty of unlabeled data and fine-tune pre-trained GNNs with few-shot labeled SDP data. Through extensive experiments conducted on the SemEval-2015 Task 18 English dataset in three formalisms (DM, PAS, and PSD), we demonstrate that our framework achieves promising results when few-shot training samples are available. Furthermore, benefiting from the pre-training process, our framework exhibits notable advantages in the out-of-domain test sets.

Keywords: Graph Contrastive Learning, Semantic Dependency Parsing, Few-Shot Learning

1. Introduction

Semantic dependency parsing (SDP) is a linguistic task that focuses on capturing intricate bi-lexical relationships, allowing words to have multiple dependency heads, and producing a labeled directed acyclic graph that accurately represents the meaning of the sentence. SDP derives from syntactic dependency parsing which aims to represent the syntactic structure of a sentence through a labeled tree. Hence, there are a lot of similarities between syntactic and semantic dependencies. SDP has been shown to be useful and widely applied in a variety of downstream tasks of natural language processing (NLP), such as sentiment analysis (Lin et al., 2019), abstractive summarization (Jin et al., 2020), natural language understanding (NLU) (Wu et al., 2021), etc.

Existing SDP models can be classified as transition-based and graph-based. Transition-based models score all transition actions according to the current parsing state and select the highest score transition action in each step. The final semantic dependency graph (SDG) could be incrementally built by a sequence of selected transition actions (Sagae and Tsujii, 2008; Tokgöz and Eryiğit, 2015; Zhang et al., 2016; Wang et al., 2018; Kurita and Søgaard, 2019; Fernández-González and Gómez-Rodríguez, 2020). Unlike the transition-based models, graph-based models score each substructure of a potential SDG and utilize exact or approximate decoding algorithms to search the highest-scoring SDG (Sun et al., 2017; Peng et al., 2017; Cao et al., 2017; Dozat and Manning, 2018; Wang et al., 2019b, 2021b; Candito, 2022). Among them, graph neural networks (GNNs) based models are especially successful because of their powerful graph representation learning ability (Li et al., 2022a,b, 2023).

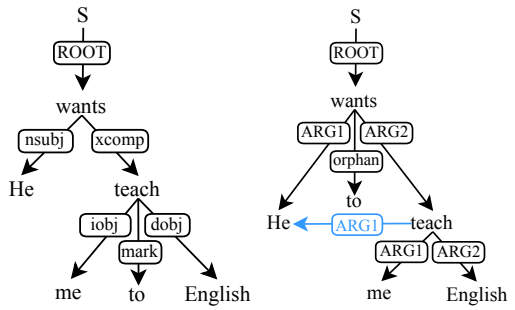
Although the benefits provided by SDP and the remarkable performance achieved by previous studies, training a high-performing SDP model requires large amounts of labeled data. This issue becomes more severe with the rise of GNNs because GNN-based models are more data-hungry and susceptible to over-fitting when lacking training data (Ju et al., 2023). To alleviate this drawback, a semi-supervised model is presented (Jia et al., 2020). This model leverages both labeled and unlabeled data to learn a dependency graph parser. Another study leverages a multitask learning framework coupled with annotation projection for languages without SDP annotated data (Aminian et al., 2020). They use annotation projection to transfer semantic annotations from a source language to the target language. These two attempts alleviate the data-hungry issue to some extent, but their performances are still not satisfactory.

Recently, contrastive learning, a category of self-supervised learning (SSL), has emerged as a new paradigm for making use of large amounts of unlabeled data when labeled data is limited (Xie et al., 2022). Contrastive learning aims to learn the representation by concentrating positive pairs and pushing negative pairs apart. Generally, contrastive learning with unlabeled data can be used as a pre-

* Corresponding Author

LREC-COLING 2024, pages 7248–7258

training process after which the limited labeled data is used to fine-tune the pre-trained deep models.



(a) Syntactic Tree (b) Semantic Graph

Figure 1: The syntactic dependency tree and the semantic dependency graph for the English sentence *He wants to teach me English*.

Furthermore, there are plenty of similarities between the syntactic dependency tree and the semantic dependency graph. For example, for the English sentence *He wants to teach me English*, its syntactic dependency tree and semantic dependency graph are shown as Figure 1. We can see that most of the dependency edges in the dependency tree and the dependency graph are the same, except for the dependency edge marked in blue.

Therefore, motivated by plenty of similarities between syntactic and semantic dependencies and the success of contrast learning in few-shot learning, we propose a syntactic dependency-guided graph contrastive learning framework for few-shot SDP (SynGCL-SDP) in this paper. There are two stages to perform few-shot SDP: the unsupervised pre-training and the supervised fine-tuning. In the pre-training stage, the plenty of unlabeled sentences will be used to construct contrastive samples and pre-train GNNs with contrastive learning. In the fine-tuning stage, the pre-trained GNNs will be fine-tuned with few-shot SDP training samples. We expect that by contrasting graphs with the similar and different structures, GNNs can be more sensitive to the structural changes. Four GNN variants, Graph Convolutional Network (GCN)(Kipf and Welling, 2016), Graph Attention Network (GAT)(Veličković et al., 2018), Graph Sample and Aggregate (GraphSage)(Hamilton et al., 2017), and Gated Graph Neural Network (GGNN)(Li et al., 2016) have been explored in Syn.

Experiments are conducted on SemEval-2015 Task 18 English dataset in three representation formalisms (DM, PAS, and PSD). Experimental results show that our framework outperforms the previous supervised and semi-supervised models when few-shot training samples are available. In addition, benefiting from the contrastive learning on a large amount of unlabeled data, our framework also perform better in the out-of-domain (OOD) test sets of three formalisms.

Contributions The contributions of our work are summarized as follows: (i) we propose a method for contrastive samples construction; (ii) we propose a syntactic dependency-guided graph contrastive learning framework to pre-train GNNs; (iii) we present pre-trained models of four GNN variants which can be used in many downstream tasks. Our code is publicly available at https://github.com/LiBinNLP/SynGCL-SDP.

2. Related Work

2.1. Semantic Dependency Parsing

Existing SDP models can be classified as transition-based and graph-based. Transition-based models utilize statistical learning approaches (e.g. structured perceptron classifier, maximum entropy classifier) or deep neural networks (e.g. Tree-LSTM, point networks) to score all transition actions according to the current parsing state and select a highest score transition action in each step. The final SDG could be incrementally built by a sequence of selected transition actions (Sagae and Tsujii, 2008; Tokgöz and Eryiğit, 2015; Zhang et al., 2016; Wang et al., 2018; Kurita and Søgaard, 2019; Fernández-González and Gómez-Rodríguez, 2020). Unlike transition-based models, graph-based models score each first-order part (independent dependency edge between two nodes) or higher-order part (combination of two or more dependency edges) of a potential SDG and utilize exact or approximate decoding algorithms to search a highest-scoring SDG (Sun et al., 2017; Peng et al., 2017; Cao et al., 2017; Dozat and Manning, 2018; Wang et al., 2019b; He and Choi, 2020; Wang et al., 2021b; Candito, 2022). Particularly, GNN-based models have made remarkable success in SDP because of their strong graph representation learning ability (Li et al., 2022a,b, 2023).

2.2. Few-Shot Semantic Dependency Parsing

Although promising performances have been achieved by previous studies, training a high-performing model requires a large amount of labeled data. To alleviate this limitation, two approaches are presented. Jia et al. (2020) presented a neural semi-supervised model which employed a conditional random field autoencoder to model the conditional reconstruction probability given the input sentence with its dependency graph as the latent variable. They leveraged both the labeled and unlabeled data to train a dependency graph parser. Aminian et al. (2020) employed a multitask

learning framework coupled with annotation projection for languages without semantically annotated data. They use annotation projection to transfer semantic annotations from a source language to a target language and use syntactic parsing as the auxiliary task to enhance the SDP model of the target language. Their work is evaluated on the generated data instead of labeled data, therefore we don't compare with it.

These two attempts alleviate the data-hungry issue to some extent, but their performances are still not satisfactory. Recently, the unsupervised contrastive pre-training and supervised fine-tuning paradigm is successful in few-shot learning. Therefore, we explore this strategy to improve few-shot SDP.

2.3. Graph Contrastive Learning

Contrastive learning is one of the categories of SSL which enables the training of deep models on unlabeled data, removing the need of excessive annotated data. Contrastive learning aims to learn the representation by concentrating positive pairs and pushing negative pairs apart.

Recently, the studies of contrastive learning have made significant progress in many NLP tasks. Wang et al. (2021a) presented a simple and effective method to automatically generate the adversarial sample and negative sample to the original sentence for pre-training a language model to enhance NLU. Das et al. (2022) presented a contrastive learning based framework that models gaussian embedding and optimizes inter token distribution distance to perform few-shot named entity recognition. Saha et al. (2022) utilized contrastive learning models that leverage simple yet efficient methods of graph perturbations to improve explanation graph generation. They transform graph generation task into a sequence-to-sequence task by graph linearization and then use contrastive learning to pre-train Transformer.

The above studies focus on pre-training a language model with contrastive sequence data. Currently, there is a trend to extend contrastive learning on graph data, namely graph contrastive learning (Xie et al., 2022). Given training graphs, graph contrastive learning aims to learn one or more encoders such that representations of similar graph instances agree with each other, and that representations of dissimilar graph instances disagree with each other. Different from the sequence-to-sequence models, we regard SDP as a sequence-to-graph task. We use graph contrastive learning to improve few-shot SDP through pre-training GNNs and fine-tuning pre-trained GNNs with limited labeled data.

3. Method

3.1. Contrastive Samples Construction

Contrastive learning is based on the use of contrastive samples, which play a key role in the discriminative learning process. Hence, it is essential to construct appropriate original, positive, and negative graphs. The details are described as follows.

Original Graph Construction To obtain the original graphs, we collect plenty of unlabeled sentences from machine translation corpus and adopt a well-trained parsing model - Stanza [1] to automatically generate a syntactic dependency tree as the original graph for each sentence. For each original graph, the data augmentation techniques are used to construct one positive graph and three negative graphs in our approach.

Positive Graph Construction Given an original graph, a positive graph will be constructed by two strategies: (1) replacing the nodes (tokens) of the original graph with synonymous words. To do so, we select words from the concept with POS tags of *Adjective, Noun, Adverb,* and *Verb* and replace them with the synonym from Wordnet (Miller, 1995) for which the cosine similarity of their word2vec representations (Mikolov et al., 2013) is the highest. (2) replacing the nodes corresponding to the recognized named entity in the sentence with the named entity of the same type in the dictionary. To do so, we collect 13 different types of named entity dictionaries (*Person, Location, Organization, Country, Artwork, Corporation, Nationality, Cardinal, Ordinal, Law, Event, Product, and Language*) and replace a named entity that appeared in the sentence with a randomly selected entity from the corresponding dictionary. It is observed that the structure of the positive graph constructed with the above two strategies has not been changed because the constraints of the dependency tree still need to be held.

Negative Graph Construction Given an original graph, three negative graphs will be constructed by following strategies: (1) deleting all correct dependency edges and randomly adding incorrect dependency edges for node pairs that have no dependency relations. (2) exchanging the head and dependency node of each dependency edge. (3) changing the dependency node to another randomly chosen node for each dependency edge.

[1] https://github.com/stanfordnlp/stanza

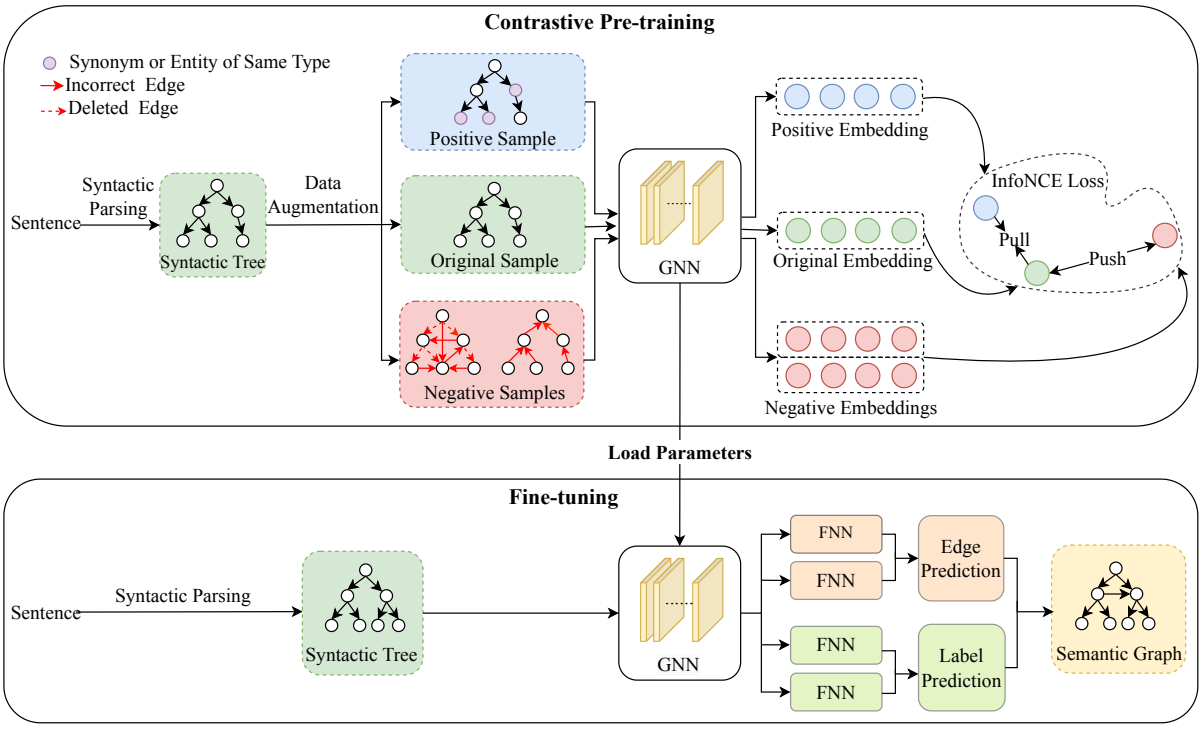


Figure 2: The overall architecture of the SynGCL-SDP. There are two stages in the SynGCL-SDP: the unsupervised pre-training on the plenty of unlabeled data and the supervised fine-tuning on the few-shot SDP data.

3.2. Contrastive Pre-training

We expect to pre-train GNNs with large amounts of unlabeled data to make them more sensitive to structural changes. To do so, contrastive learning is employed to make the original samples closer to the positive samples and further away from the negative samples.

Contextualized Representation Learning For a sentence with n words, we concatenate word and feature embeddings, and feed them into a Bi-directional Long Short-Term Memory (BiLSTM) to obtain the contextualized representation of each word, as Eq. 1 and Eq. 2:

$$x_i = e_i^{(word)} \oplus e_i^{(tag)} \oplus e_i^{(lemma)} \oplus e_i^{(char)} \quad (1)$$

$$c_i = BiLSTM(x_i) \quad (2)$$

where $e_i^{(word)}$ denotes the word embedding of word w_i, $e_i^{(tag)}$, $e_i^{(lemma)}$, and $e_i^{(char)}$ denote POS (part-of-speech) tag embedding, lemma embedding and character embedding that is generated by Char-LSTM (Kim et al., 2016), x_i is the concatenation ($\oplus$) of the word and feature embeddings of word w_i, c_i is the contextualized representation of w_i. Stacking c_i for $i = 0, 1, \ldots, n$ forms the node feature matrix C.

Graph Representation Learning GNNs encode node embeddings in a similar incremental manner: one GNN layer encodes information about immediate neighbors and K layers encode K-order neighborhoods (i.e., information about nodes at most K hops away). K-layer GNNs are utilized to learn the graph representation (i.e. node embedding) for each node. GNNs take in the node feature matrix C and the adjacency matrix A and output the embedding matrix of the final layer as node embeddings H.

The node embedding matrix H is computed in a similar incremental manner, as Eq. 3:

$$H^{(k)} = \textit{GNNLayer}^{(k-1)}(H^{(k-1)}, A) \quad (3)$$

where $H^{(k)}$ is the node embedding matrix of k^{th} layer, $H^{(0)} = C$, $\textit{GNNLayer}^{(k)}$ denotes the k^{th} GNN layer, adjacency matrix A is extracted from the corresponding syntactic dependency tree.

Contrastive Embedding The final node representation z_i is the concatenation of the contextualized representation c_i and graph representation h_i, as Eq. 4:

$$z_i = c_i \oplus h_i \quad (4)$$

The paired original sample, positive sample, and negative samples will be processed as Eq. 1 – Eq. 4, to obtain the paired original embedding, positive embedding, and negative embeddings.

Pre-training Objective InfoNCE contrastive loss function (Oord et al., 2018) is used as the pre-training objective, as Eq. 5:

$$\mathcal{L}_{pt} = -\log \frac{exp(sim(z, z^+))/\tau}{\sum_{z^- \in Z^-} exp(sim(z, z^-))/\tau} \quad (5)$$

where z denotes the original embedding, z^+ denotes the positive embedding, z^- denotes the negative embedding, Z^- denotes the collection of all negative embeddings, τ is the temperature and $sim(.,.)$ is the cosine similarity function that measures the similarity of contrastive embeddings. The Adaptive Moment Estimation (Adam) (Kingma and Ba, 2015) method is used to optimize the contrastive pre-training loss $\mathcal{L}_{pt}$, and the exponential decay strategy is used for annealing the learning rate.

3.3. Fine-tuning

In the fine-tuning stage, the contrastive pre-trained GNNs are then used as the initialization of the GNNs in a supervised schema. A sentence s with n words will be processed as Eq. 1 – Eq. 4 to get the final node representation z_i for each word w_i.

Then the four feed-forward neural networks (FNN) are used to get different representations in four aspects, as Eq. 6 – Eq. 9:

$$h_i^{(edge-head)} = \mathit{FNN}^{(edge-head)}(z_i) \quad (6)$$

$$h_i^{(label-head)} = \mathit{FNN}^{(label-head)}(z_i) \quad (7)$$

$$h_i^{(edge-dep)} = \mathit{FNN}^{(edge-dep)}(z_i) \quad (8)$$

$$h_i^{(label-dep)} = \mathit{FNN}^{(label-dep)}(z_i) \quad (9)$$

Two biaffine classifiers are used to predict edges and labels, as Equation 10 and 11 :

$$s_{i,j}^{(edge)} = Biaff^{(edge)}(h_i^{(edge-dep)}, h_j^{(edge-head)}) \quad (10)$$

$$s_{i,j}^{(label)} = Biaff^{(label)}(h_i^{(label-dep)}, h_j^{(label-head)}) \quad (11)$$

where $s_{i,j}^{(edge)}$ and $s_{i,j}^{(label)}$ are scores of the edge and label between the word w_i and w_j.

$s_{i,j}^{(edge)}$ is a scalar and $s_{i,j}^{(label)}$ is a vector. There is a dependency edge between the word w_i and w_j where $s_{i,j}^{(edge)}$ is positive, as Eq. 12; the most probable label will be assigned to the edge, as Eq. 13.

$$\hat{y}_{i,j}^{(edge)} = \{s_{i,j}^{(edge)} > 0\} \quad (12)$$

$$\hat{y}_{i,j}^{(label)} = \arg\max s_{i,j}^{(label)} \quad (13)$$

Fine-tuning Objective We define the loss function of the edge prediction module (as Eq. 14) and the label prediction module (as Eq. 15):

$$\mathcal{L}^{(edge)}(\theta_1) = CE(\hat{y}_{i,j}^{(edge)}, y_{i,j}^{(edge)}) \quad (14)$$

$$\mathcal{L}^{(label)}(\theta_2) = CE(\hat{y}_{i,j}^{(label)}, y_{i,j}^{(label)}) \quad (15)$$

where θ_1, θ_2 are the parameters of two modules, $CE(.,.)$ is cross entropy loss function.

We can train the system by summing the losses of two modules, and back propagating error to the parser. Then the Adam (Kingma and Ba, 2015) method is used to optimize the summed fine-tuning loss function $\mathcal{L}_{ft}$, and the exponential decay strategy is used for annealing the learning rate:

$$\mathcal{L}_{ft} = \alpha\mathcal{L}^{(edge)} + (1-\alpha)\mathcal{L}^{(label)} \quad (16)$$

where $\alpha \in (0, 1)$ is a tunable interpolation constant.

4. Experiment

4.1. Settings

Dataset The experiments are conducted on the SemEval-2015 Task 18 (Oepen et al., 2015) English dataset in three representation formalisms: DELPH-IN MRS (DM) (Flickinger et al., 2012), Predicate-Argument Structure (PAS) (Miyao and Tsujii, 2004), and Prague Semantic Dependencies (PSD) (Hajic et al., 2012). 33,964 sentences from Sections 00-19 of the Wall Street Journal corpus as the training set, 1,692 sentences from Section 20 as development set, 1,410 sentences from Section 21 as in-domain (ID) test set, and 1,849 sentences sampled from the Brown Corpus as the out-of-domain (OOD) test set.

Unlabeled Data for Contrastive Pre-training In the contrastive pre-training stage, the unlabeled raw sentences are downloaded from the WMT14 machine translation monolingual training data[2]. Considering that the scale of GNN's parameters is not large, we randomly select 50,000 raw sentences for constructing the contrastive samples.

Few-Shot Data Sampling To evaluate the proposed model on the few-shot labeled SDP data, we randomly sample the few-shot training data from the training set with five different sampling rates (i.e., percentage of labeled data): 1%, 10%, 30%, 50%, and 100%. The sampling procedure is carried out three times to get the experimental results as accurate and fair as possible.

[2] http://statmt.org/wmt14/training-monolingual-news-crawl/news.2010.en.shuffled.gz

Form		Models	Percentage of Labeled Data									
			1%		10%		30%		50%		100%	
			UF1	LF1	UF1	LF1	UF1	LF1	UF1	LF1	UF1	LF1
DM	ID	Biaffine (2018)	77.11	75.07	87.19	86.95	91.79	91.02	92.94	92.36	94.33	93.73
		Semi-SDP (2020)	77.90	75.49	87.82	86.60	92.50	91.67	93.74	93.04	95.06	93.93
		DynGL-SDP (2022b)	77.27	75.05	88.05	86.97	92.97	92.21	94.00	93.22	**95.25**	**94.85**
		SynGCL-SDP (GGNN)	**81.45**†	**78.77**†	**90.82**†	**89.59**†	**93.23**†	**92.34**†	**94.16**†	**93.40**†	95.04	94.42
	OOD	Biaffine (2018)	70.97	67.31	81.15	79.04	86.88	85.60	88.34	87.75	90.43	89.16
		Semi-SDP (2020)	72.62	69.61	82.68	80.77	87.70	86.30	89.36	88.14	91.04	90.08
		DynGL-SDP (2022b)	72.78	69.91	82.92	82.65	87.17	86.74	89.12	87.81	**91.64**	**90.73**
		SynGCL-SDP (GGNN)	**76.53**†	**73.13**†	**86.03**†	**84.12**†	**88.97**†	**87.55**†	**90.12**†	**88.88**†	91.33	90.25
PAS	ID	Biaffine (2018)	82.39	80.87	90.52	89.96	93.40	92.74	94.11	93.39	94.65	94.01
		Semi-SDP (2020)	83.14	81.61	91.43	90.55	94.11	93.32	94.71	94.06	95.52	94.91
		DynGL-SDP (2022b)	81.69	80.26	90.63	89.81	94.11	93.37	94.79	94.14	**95.76**	**95.12**
		SynGCL-SDP (GGNN)	**86.01**†	**84.23**†	**92.84**†	**91.95**†	**94.40**†	**93.71**†	**95.02**†	**94.31**†	95.66	95.04
	OOD	Biaffine (2018)	76.48	74.40	85.52	84.33	89.69	88.81	91.02	89.97	91.69	90.88
		Semi-SDP (2020)	77.09	75.00	86.31	85.01	90.40	89.30	91.63	90.52	**92.65**	**91.73**
		DynGL-SDP (2022b)	75.90	73.79	85.30	84.05	90.35	84.21	91.52	90.46	92.23	91.31
		SynGCL-SDP (GGNN)	**79.80**†	**77.15**†	**87.57**†	**86.11**†	**90.69**†	**89.42**†	**91.91**†	**90.87**†	92.59	91.55
PSD	ID	Biaffine (2018)	75.92	67.25	86.69	78.23	91.37	83.58	91.92	84.13	92.58	84.41
		Semi-SDP (2020)	76.47	67.72	87.10	78.87	91.43	83.62	92.46	84.56	93.78	86.63
		DynGL-SDP (2022b)	77.45	67.51	88.75	79.08	91.79	83.48	92.15	**84.93**	**93.73**	**86.60**
		SynGCL-SDP (GGNN)	**82.09**†	**70.55**†	**89.70**†	**79.94**†	**91.42**†	**83.66**†	**92.57**†	84.31	93.53	86.09
	OOD	Biaffine (2018)	76.94	68.64	85.51	78.08	89.25	81.86	90.53	83.65	88.69	81.44
		Semi-SDP (2020)	77.05	68.83	85.89	78.20	89.32	81.88	90.70	**83.66**	**91.87**	**85.24**
		DynGL-SDP (2022b)	78.39	69.23	85.74	77.26	89.71	81.97	90.22	83.58	91.54	84.98
		SynGCL-SDP (GGNN)	**81.96**†	**70.45**†	**88.74**†	**79.31**†	**90.35**†	**82.03**†	**91.08**†	83.47	91.82	84.39

Table 1: Experimental results with varying percentages of labeled data on SemEval-2015 Task 18 English dataset in three formalisms. The result on each few-shot training set is averaged over 3 runs. To make a fair comparison, the results of the three compared approaches are reproduced on the sampled few-shot data. ID denotes the in-domain test set, and OOD denotes the out-of-domain test set. The bold number indicates the highest score in the corresponding settings. † means that the score outperforms compared approaches with a significance level of 0.05 by means of the paired student's t-test.

Evaluation Metrics Unlabeled F1 score (UF1) and Labeled F1 score (LF1) are used as the metrics to evaluate the performance of compared approaches and our model on the ID and OOD test sets for each formalism. The reported score of each model in each formalism is averaged over three runs (each time with a new randomly sampled few-shot training set).

4.2. Compared Approaches

We compare the proposed model SynGCL-SDP (the GNN module of SynGCL-SDP is implemented with GGNN) with previous approaches: (1) Biaffine (Dozat and Manning, 2018) is a simple but accurate supervised model. (2) Semi-SDP (Jia et al., 2020) is a semi-supervised model which aims to improve performance with both the labeled and unlabeled data. (3) DynGL-SDP (Li et al., 2022b) is a dynamic graph learning-based model, which also achieves the start-of-the-art (SOTA) performance. Three types of features (tag, char, lemma) are used in our model and compared approaches.

4.3. Hyperparameters

The hyperparameter configuration for our final system is given in Table 2. 100-dimensional pretrained GloVe(Pennington et al., 2014) embeddings are used for English. Only words or lemmas that occurred 7 times or more will be included in the word and lemma embedding matrix.

4.4. Main Results

Experimental results on English dataset in three formalisms are shown as Table 1. From the main results, we can see that our proposed model performs better than the compared models on most few-shot data groups, especially with the 1% labeled data (only 339 labeled sentences are used). This highly suggests that our model is superior in

Layer	Hyperparameter	Value
Word Embed	English	125
Feature Embed	POS/Char/Lemma	100
LSTM	layers	3
	hidden size	300
	dropout	0.33
GNN	layers	3
	hidden	600
	dropout	0.25
	GAT heads	3
MLP	edge-head/dep hidden	600
	label-head/dep hidden	600
	dropout	0.25
Pre-training	epochs	100
	optimizer	Adam
	learning rate	$1e^{-3}$
	Adam (β_1, β_2)	(0.0, 0.95)
	decay rate	0.75
	Loss τ	0.1
Fine-tunning	epochs	150
	optimizer	Adam
	learning rate	$1e^{-3}$
	Adam (β_1, β_2)	(0.0, 0.95)
	decay rate	0.75
	Loss α	0.1

Table 2: Final hyperparameter configuration.

few-shot SDP. Particularly, benefiting from the pre-training stage on plenty of the unlabeled sentences, our model shows more advantages on the OOD test sets, which suggests the good generalization of our model.

As the percentage of the labeled data increases, the performances of all models goes up, but the advantage of our model decreases. The reason is that the compared neural models can also perform well when the plenty of the labeled data is available. Particularly, when using all the training data, our model performs worse than DynGL-SDP which is a dynamic graph learning-based model. The reason for this result is because the SynGCL-SDP learns the graph representation based on the static graph (i.e., syntactic dependency tree) which is constructed by an existing parser (stanza), the noise in the static graph will be accumulated and propagated to the later stages. Combining the graph contrastive learning and the dynamic graph learning may be helpful.

Furthermore, the UF1 and LF1 of all models on the PSD representation formalism have a big gap because the number of dependency labels of PSD formalism is much larger than the other two representation formalisms (DM and PAS), making the dependency labels harder to be predicted. We think that using the heterogeneous GNNs like HGCN(Yang et al., 2021) and HAN(Wang et al., 2019a) to model the dependency labels of the contrastive graphs might alleviate this problem. We will explore it in our future study.

4.5. Effect of Pre-training and Features

Our model significantly improves the performance of few-shot SDP by pre-training GNNs with plenty of unlabeled data and fine-tuning on few-shot labeled samples when the three types of feature embeddings (as Eq.1) are used. We also notice that the previous studies (Li et al., 2022a,b) have shown that GNNs are helpful for improving the performance of SDP without pre-training.

To investigate the effect of pre-training stage and the effect of each type of feature embedding, we conduct a controlled experiment on the SemEval-2015 Task 18 English dataset in DM formalism with the combination of three types of features, in which the one loads the pre-trained GNNs for initialization (SynGCL-SDP) and another not (SynGCL-SDP$^{\times}$). The result is shown as Table 3. We can see that the performance of the model that uses the pre-trained GNNs for initialization outperforms the model that doesn't use the pre-trained GNNs in all few-shot sampling groups and all combinations of the three types of features. Moreover, with the increase of the labeled SDP data, the advantage of SynGCL-SDP gradually decreases.

4.6. Effect of Different GNN Variants

In the experiment, the GNN module of SynGCL-SDP is implemented with GGNN. Besides the GGNN, there are several GNN variants have been presented and commonly used. To fully explore the capabilities of our model, we compare the effect of different GNNs by replacing GNN module of SynGCL-SDP with the corresponding GNN variant.

Table 4 shows the performances of SynGCL-SDP using four GNN variants (GCN, GAT, GraphSage, and GGNN) on the English dataset in DM formalism. Compared to the previous supervised studies, SynGCL-SDP using four GNN variants perform better with few-shot labeled SDP data, especially using GAT, GGNN, and GraphSage.

4.7. Convergence Behavior

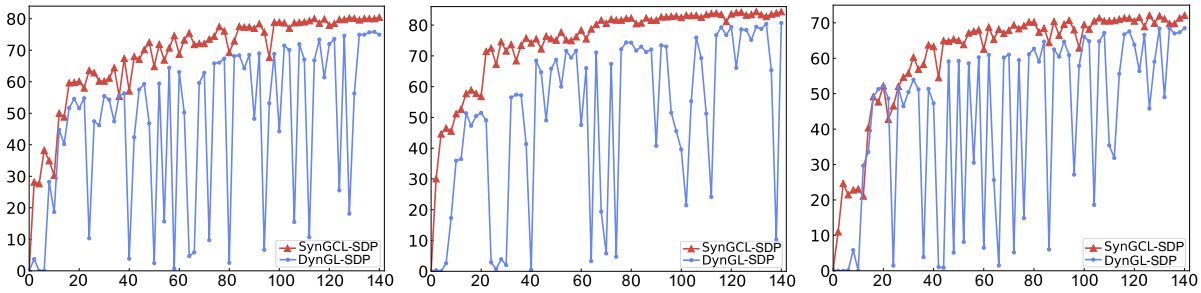
Figure 3 compares the convergence curves of our model (SynGCL-SDP) and the GNN-based SOTA model (DynGL-SDP) when using 1% training data. We plot one data point corresponding to the LF1 every 2 epochs for clarity. From the compared curves, we can clearly see that the performance

Feature		Models	Percentage of Labeled Data 1% UF1	1% LF1	10% UF1	10% LF1	30% UF1	30% LF1	50% UF1	50% LF1	100% UF1	100% LF1
tag	ID	SynGCL-SDP$^\flat$	80.41	77.23	89.44	88.02	92.20	91.21	93.09	92.12	94.20	93.42
		SynGCL-SDP	**80.88**	**77.98**	**90.12**	**88.56**	**92.49**	**91.49**	**93.32**	**92.15**	**94.31**	**93.45**
	OOD	SynGCL-SDP$^\flat$	74.63	71.13	84.47	82.19	87.78	86.15	88.59	87.32	90.17	88.80
		SynGCL-SDP	**76.26**	**72.87**	**84.73**	**82.63**	**88.11**	**86.38**	**88.72**	**87.51**	**90.41**	**89.08**
tag+char	ID	SynGCL-SDP$^\flat$	80.87	78.30	89.73	88.45	92.50	91.63	93.25	92.41	94.30	93.59
		SynGCL-SDP	**81.27**	**78.59**	**90.31**	**88.97**	**92.78**	**91.83**	**93.54**	**92.54**	**94.50**	**93.66**
	OOD	SynGCL-SDP$^\flat$	74.51	71.50	84.46	82.56	87.99	86.42	89.07	87.70	90.50	89.31
		SynGCL-SDP	**77.16**	**73.88**	**84.49**	**82.63**	**88.16**	**86.64**	**89.53**	**88.11**	**90.51**	**89.35**
tag+char+lemma	ID	SynGCL-SDP$^\flat$	80.83	77.70	90.55	89.38	93.17	92.32	94.06	93.34	94.98	94.40
		SynGCL-SDP	**81.45**	**78.77**	**90.82**	**89.59**	**93.23**	**92.34**	**94.16**	**93.40**	**95.04**	**94.42**
	OOD	SynGCL-SDP$^\flat$	75.82	72.51	85.64	83.77	88.70	87.31	90.02	88.81	91.24	90.19
		SynGCL-SDP	**76.53**	**73.13**	**86.03**	**84.12**	**88.97**	**87.55**	**90.12**	**88.88**	**91.33**	**90.25**

Table 3: The UF1 and LF1 of English dataset in DM formalism for two models in which the one loads the pre-trained GNNs for initialization (SynGCL-SDP) and another not (SynGCL-SDP$^\flat$). The GNN modules of them are implemented with GGNN. Two models are augmented with the combinations of three types of features (tag, char, lemma denote part-of-speech tag embedding, character embedding and lemma embedding respectively).

	Models	Percentage of Labeled Data 1% UF1	1% LF1	10% UF1	10% LF1	30% UF1	30% LF1	50% UF1	50% LF1	100% UF1	100% LF1
ID	SynGCL-SDP(GCN)	80.78	77.98	90.19	88.87	92.92	91.94	93.99	93.17	94.93	94.23
	SynGCL-SDP(GAT)	80.58	77.84	90.24	88.97	**93.23**	**92.37**	94.06	93.31	94.87	94.26
	SynGCL-SDP(GGNN)	**81.45**	**78.77**	**90.82**	**89.59**	**93.23**	92.34	94.16	93.40	95.04	94.42
	SynGCL-SDP(GraphSage)	81.35	78.64	90.76	89.57	93.17	92.31	**94.18**	**93.45**	**95.24**	**94.62**
OOD	SynGCL-SDP(GCN)	75.60	72.02	85.51	83.52	88.62	87.15	89.84	88.54	91.24	90.15
	SynGCL-SDP(GAT)	**77.46**	**74.06**	85.72	83.82	**89.02**	**87.57**	89.96	88.70	91.31	90.17
	SynGCL-SDP(GGNN)	76.53	73.13	**86.03**	**84.12**	88.97	87.55	**90.12**	**88.88**	91.33	90.25
	SynGCL-SDP(GraphSage)	76.39	72.97	85.82	83.98	88.58	87.12	90.09	88.87	**91.35**	**90.31**

Table 4: The UF1 and LF1 of SynGCL-SDP on the English dataset in DM formalism with three types of features, in which the GNN module is implemented with four pre-trained GNN variants. Each score is averaged over 3 runs.



(a) DM (b) PAS (c) PSD

Figure 3: Convergence curves (LF1 vs. training epochs) of SynGCL-SDP and DynGL-SDP on dev data in three formalisms (DM, PAS, and PSD) when using 1% training data.

of DynGL-SDP is quite unstable during the training process when there are very few labeled samples available, meaning that it is susceptible to overfitting. On the contrary, the performance of our

model improves steadily until it converges as the number of training epochs increases, indicating that our model is still stable and not prone to over-fitting when few labeled data is available.

5. Conclusion

In this paper, we propose a syntax-guided graph contrastive learning framework for few-shot SDP. The proposed framework pre-trains GNNs with plenty of unlabeled data and fine-tunes the pre-trained GNNs with few-shot labeled SDP data. The pretrained GNNs can also take advantage of large amounts of unlabeled data to adapt to out-of-domain. Extensive evaluations on SemEval-2015 Task 18 English dataset in three formalisms show that our model performs better when limited data is available.

Maryam Aminian, Mohammad Sadegh Rasooli, and Mona Diab. 2020. Multitask learning for cross-lingual transfer of broad-coverage semantic dependencies. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8268–8274.

Marie Candito. 2022. Auxiliary tasks to boost biaffine semantic dependency parsing. In *Findings of the Association for Computational Linguistics: ACL 2022*.

Junjie Cao, Sheng Huang, Weiwei Sun, and Xiaojun Wan. 2017. Quasi-second-order parsing for 1-endpoint-crossing, pagenumber-2 graphs. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 24–34.

Sarkar Snigdha Sarathi Das, Arzoo Katiyar, Rebecca J Passonneau, and Rui Zhang. 2022. Container: Few-shot named entity recognition via contrastive learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6338–6353.

Timothy Dozat and Christopher D Manning. 2018. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020. Transition-based semantic dependency parsing with pointer networks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7035–7046.

Dan Flickinger, Yi Zhang, and Valia Kordoni. 2012. Deepbank. a dynamically annotated treebank of the wall street journal. In *Proceedings of the 11th International Workshop on Treebanks and Linguistic Theories*, pages 85–96.

Jan Hajic, Eva Hajicová, Jarmila Panevová, Petr Sgall, Ondřej Bojar, Silvie Cinková, Eva Fucíková, Marie Mikulová, Petr Pajas, Jan Popelka, et al. 2012. Announcing prague czech-english dependency treebank 2.0. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3153–3160.

William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035.

Han He and Jinho Choi. 2020. Establishing strong baselines for the new decade: Sequence tagging, syntactic and semantic parsing with bert. In *The Thirty-Third International Flairs Conference*.

Zixia Jia, Youmi Ma, Jiong Cai, and Kewei Tu. 2020. Semi-supervised semantic dependency parsing using crf autoencoders. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6795–6805.

Hanqi Jin, Tianming Wang, and Xiaojun Wan. 2020. Semsum: Semantic dependency guided neural abstractive summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8026–8033.

Wei Ju, Zequn Liu, Yifang Qin, Bin Feng, Chen Wang, Zhihui Guo, Xiao Luo, and Ming Zhang. 2023. Few-shot molecular property prediction via hierarchically structured learning on relation graphs. *Neural Networks*, 163:122–131.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Thirtieth AAAI conference on artificial intelligence*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Shuhei Kurita and Anders Søgaard. 2019. Multitask semantic dependency parsing with policy

gradient for learning easy-first strategies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2420–2430.

Bin Li, Yunlong Fan, Miao Gao, Yikemaiti Sataer, and Zhiqiang Gao. 2023. A joint-learning-based dynamic graph learning framework for structured prediction. *Electronics*, 12(11).

Bin Li, Yunlong Fan, Yikemaiti Sataer, Zhiqiang Gao, and Yaocheng Gui. 2022a. Improving semantic dependency parsing with higher-order information encoded by graph neural networks. *Applied Sciences*, 12(8).

Bin Li, Miao Gao, Yunlong Fan, Yikemaiti Sataer, Zhiqiang Gao, and Yaocheng Gui. 2022b. DynGL-SDP: Dynamic graph learning for semantic dependency parsing. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3994–4004, Gyeongju, Republic of Korea.

Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. 2016. Gated graph sequence neural networks. In *Proceedings of 4th International Conference on Learning Representations (ICLR 2016)*.

Peiqin Lin, Meng Yang, and Jianhuang Lai. 2019. Deep mask memory network with semantic dependency and context moment for aspect level sentiment classification. In *Proceedings of the 2019 International Joint Conferences on Artificial Intelligence*, pages 5088–5094.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Yusuke Miyao and Jun'ichi Tsujii. 2004. Deep linguistic analysis for the accurate identification of predicate-argument relations. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pages 1392–1398.

Stephan Oepen, Marco Kuhlmann, Yusuke Miyao, Daniel Zeman, Silvie Cinková, Dan Flickinger, Jan Hajic, and Zdenka Uresova. 2015. Semeval 2015 task 18: Broad-coverage semantic dependency parsing. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 915–926.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Hao Peng, Sam Thomson, and Noah A Smith. 2017. Deep multitask learning for semantic dependency parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2037–2048.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency dag parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 753–760.

Swarnadeep Saha, Prateek Yadav, and Mohit Bansal. 2022. Explanation graph generation via pre-trained language models: An empirical study with contrastive learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1190–1208.

Weiwei Sun, Junjie Cao, and Xiaojun Wan. 2017. Semantic dependency parsing via book embedding. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 828–838.

Alper Tokgöz and Gülşen Eryiğit. 2015. Transition-based dependency dag parsing using dynamic oracles. In *Proceedings of the ACL-IJCNLP 2015 Student Research Workshop*, pages 22–27.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Dong Wang, Ning Ding, Piji Li, and Haitao Zheng. 2021a. Cline: Contrastive learning with semantic negative examples for natural language understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2332–2342.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019a. Heterogeneous graph attention network. In *The world wide web conference*, pages 2022–2032.

Xinyu Wang, Jingxian Huang, and Kewei Tu. 2019b. Second-order semantic dependency parsing with end-to-end neural networks. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4618.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021b. Automated concatenation of embeddings for structured prediction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2643–2660.

Yuxuan Wang, Wanxiang Che, Jiang Guo, and Ting Liu. 2018. A neural transition-based approach for semantic dependency graph parsing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Zhaofeng Wu, Hao Peng, and Noah A Smith. 2021. Infusing finetuning with semantic dependencies. *Transactions of the Association for Computational Linguistics*, 9:226–242.

Yaochen Xie, Zhao Xu, Jingtun Zhang, Zhengyang Wang, and Shuiwang Ji. 2022. Self-supervised learning of graph neural networks: A unified review. *IEEE transactions on pattern analysis and machine intelligence*.

Yaming Yang, Ziyu Guan, Jianxin Li, Wei Zhao, Jiangtao Cui, and Quan Wang. 2021. Interpretable and efficient heterogeneous graph convolutional network. *IEEE Transactions on Knowledge and Data Engineering*.

Xun Zhang, Yantao Du, Weiwei Sun, and Xiaojun Wan. 2016. Transition-based parsing for deep dependency structures. *Computational Linguistics*, 42(3):353–389.