Transferability of Syntax-Aware Graph Neural Networks in Zero-Shot Cross-Lingual Semantic Role Labeling

Rachel Sidney Devianti
Department of Computer Science
The University of Tokyo
sidneyrachel@is.s.u-tokyo.ac.jp

Yusuke Miyao
Department of Computer Science
The University of Tokyo
yusuke@is.s.u-tokyo.ac.jp

Abstract

Recent models in cross-lingual semantic role labeling (SRL) barely analyze the applicability of their network selection. We believe that network selection is important since it affects the transferability of cross-lingual models, i.e., how the model can extract universal features from source languages to label target languages. Therefore, we comprehensively compare the transferability of different graph neural network (GNN)-based models enriched with universal dependency trees. GNN-based models include transformer-based, graph convolutional network-based, and graph attention network (GAT)-based models. We focus our study on a zero-shot setting by training the models in English and evaluating the models in 23 target languages provided by the Universal Proposition Bank. Based on our experiments, we consistently show that syntax from universal dependency trees is essential for cross-lingual SRL models to achieve better transferability. Dependency-aware self-attention with relative position representations (SAN-RPRs) transfer best across languages, especially in the long-range dependency distance. We also show that dependency-aware two-attention relational GATs transfer better than SAN-RPRs in languages where most arguments lie in a 1-2 dependency distance.

1 Introduction

Semantic role labeling (SRL) is a task to assign semantic roles to words or phrases in a sentence concerning a specific predicate, as shown in Figure 1. SRL supports many natural language processing (NLP) tasks, e.g., information extraction (Christensen et al., 2010; Stanovsky and Dagan, 2016), abstractive summarization (Khan et al., 2015), and machine translation (Rapp, 2022). However, SRL resource availability is still low, hindering the performance of other NLP tasks in diverse languages. Cross-lingual approaches try to solve this problem by enriching the models with knowledge from resource-rich languages (source languages) so that they perform better in resource-poor languages (target languages).

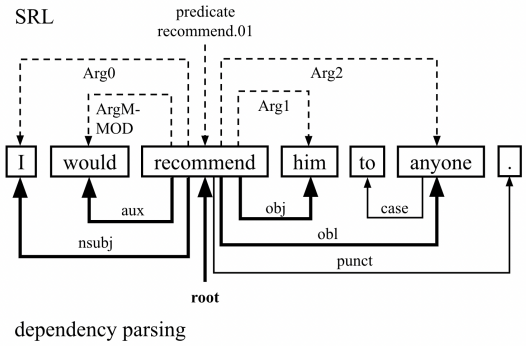


Figure 1: An example of SRL task (top) and dependency parsing task (bottom) applied to a sentence taken from Universal Proposition Bank (UPB). The bold lines indicate path co-occurrences in both tasks.

Existing studies in cross-lingual SRL improve existing models by (1) separating language-universal and language-specific components (Fei et al., 2020; Conia et al., 2021) to capture both language-universal and language-specific features from each language and (2) minimizing the dependence on external tools, e.g., word alignment and machine translation tools (Cai and Lapata, 2020; Conia et al., 2021). Nevertheless, existing studies in cross-lingual SRL barely analyze the network selection's applicability, which affects the models' transferability.

Therefore, in this work, we comprehensively compare diverse graph neural networks (GNNs) to investigate their transferability in cross-lingual SRL. Various GNNs have been shown to be useful to encode and transfer linguistic structures in similar tasks cross-lingually. Furthermore, we encode universal dependency trees in the networks, for the following reasons: (1) Many predicate-argument paths and argument roles in SRL co-occur with dependency paths and dependency relations in depen-

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 20–42

dency parsing (Marcheggiani and Titov, 2017) (Figure 1). (2) A universal dependency tree represents a sentence's grammatical structure in a language-universal scheme. (3) Universal dependency trees help cross-lingual models achieve better transferability (Ahmad et al., 2021a; Ahmad et al., 2021b; Zhang et al., 2021).

We conduct comprehensive experiments on various networks as encoders, including transformer-based, graph convolutional network (GCN)-based, and graph attention network (GAT)-based encoders, which we categorize as GNN-based encoders. We also compare the results with a BiLSTM-based encoder as a baseline. Transformer-based models have been proven effective in performing cross-lingually in dependency parsing (Ahmad et al., 2019) and event argument role labeling (EARL) (Ahmad et al., 2021b). Furthermore, we also investigate GCN-based and GAT-based models because different NLP tasks, e.g., monolingual SRL (Marcheggiani and Titov, 2017), cross-lingual SRL (Zhang et al., 2021), aspect-based sentiment analysis (ABSA) (Wang et al., 2020; Jiang et al., 2021), and relation prediction (Nathani et al., 2019), have shown the effectiveness of exploiting the networks to encode dependency trees in their models.

Following previous work (Fei et al., 2020), we limit our exploration to argument detection and argument labeling in the dependency-based SRL. We conduct experiments in a zero-shot setting to find the most transferable network across languages. We train and evaluate the models in 23 languages provided by Universal Proposition Bank (UPB) v2. We show that: (1) Universal dependency trees are essential for cross-lingual SRL models to achieve better transferability. (2) A dependency-aware transformer-based model, i.e., SAN-RPRs, outperforms other models, especially as the dependency distance increases. (3) A dependency-aware GAT-based model, i.e., two-attention relational GATs (TAGATs), transfer best in languages where most arguments lie in 1-2 dependency distance. In addition, our findings can also shed some light on selecting the most appropriate network for building cross-lingual models in similar tasks.

2 Background

2.1 Universal Proposition Bank

Universal Proposition Bank (UPB) is a corpus containing SRL annotations for diverse languages. UPB v2 (Jindal et al., 2022) provides SRL annotations for 44 treebanks consisting of 24 languages, including English as our source language, as shown in Table 1. UPB is annotated semi-automatically through filtered annotation projection and bootstrap training (Akbik et al., 2015). UPB v2 has significantly improved over UPB v1 regarding SRL annotation quality, language scope, and availability of span-based SRL annotations (Jindal et al., 2022). We use dependency-based SRL annotations in UPB v2 that are annotated on top of UD v2.9 throughout our experiments.

Languages in UPB v2		
English (EN)	Czech (CS)	Dutch (NL)
Chinese (ZH)	Greek (EL)	Polish (PL)
Finnish (FI)	Korean (KO)	Telugu (TE)
Italian (IT)	Romanian (RO)	Indonesian (ID)
Spanish (ES)	Hindi (HI)	Japanese (JA)
French (FR)	Marathi (MR)	Russian (RU)
German (DE)	Tamil (TA)	Ukrainian (UK)
Portuguese (PT)	Hungarian (HU)	Vietnamese (VI)

Table 1: The list of languages in UPB v2.

2.2 Universal Dependencies

Universal Dependencies (UD) is a corpus containing consistent syntactic annotations for diverse languages, such as part-of-speech (POS) tags, morphological features, and dependency tree annotations. UD v1 (Nivre et al., 2016) and UD v2 (Nivre et al., 2020) have different annotation schemes[1] in terms of word segmentation, POS tags, morphological features, and syntactic relations. UD v1 and UD v2 have 40[2] and 37[3] universal dependency relations, respectively. UD v2.9 contains dependency tree annotations for 217 treebanks of 122 languages.

2.3 Dependency-based Semantic Role Labeling

Instead of labeling the whole argument span with a semantic role, dependency-based SRL only labels the argument head, i.e., the head of the argument span according to the dependency tree (Surdeanu et al., 2008). For example, in Figure 1, the phrase "to anyone" is the argument "ARG2" of the predicate "recommend". Based on the dependency tree at the bottom of the figure, "anyone" is the head of the phrase "to anyone". Therefore, dependency-based SRL annotates the edge that connects "recommend" to "anyone" with the argument "ARG2".

[1] https://universaldependencies.org/v2/summary.html
[2] https://universaldependencies.org/docsv1/u/dep/
[3] https://universaldependencies.org/u/dep/

2.4 Related Work

Existing cross-lingual SRL models (Fei et al., 2020; Cai and Lapata, 2020; Conia et al., 2021) utilize BiLSTM-based models to encode sentences sequentially, even though LSTMs do not transfer effectively across languages (Ahmad et al., 2019). On the other hand, GNNs have been used to encode dependency trees in building models for different NLP tasks, e.g., monolingual SRL, aspect-based sentiment analysis (ABSA), event argument role labeling (EARL), and relation prediction. In monolingual SRL, Marcheggiani and Titov (2017) employ syntactic GCNs (SGCNs) on top of BiLSTMs to incorporate dependency trees as graphs. In ABSA, Wang et al. (2020) and Jiang et al. (2021) apply relational GATs (R-GATs) and attention-based relational GCNs (ARGCNs), respectively, over modified dependency trees to establish direct connections between aspects and their corresponding opinion words. In EARL, Ahmad et al. (2021b) modify vanilla Transformers to encode syntactic structures from dependency trees, i.e., graph attention transformer encoders (GATEs). Finally, Nathani et al. (2019) propose KBGATs as a modification to original GATs (Veličković et al., 2018) to encode nodes and edge relations for relation prediction in knowledge graphs.

3 Models

We apply a standard encoder-decoder architecture to compare transformer-based, GCN-based, GAT-based, and BiLSTM-based cross-lingual SRL models. Section 3.1 describes a method to convert a dependency tree into a graph for building graph neural networks (GNNs). Section 3.2, 3.3, and 3.4 describe the input layer, encoder, and decoder of the architecture. The implementation of the model architecture is available on `github.com/mynlp/syntax-aware-cross-lingual-srl`.

3.1 Graph Construction

We explain how to construct dependency relation representation (DR) as the edge representation of the graphs in GNNs. DR encodes the dependency direction (i.e., self-connection, head-to-dependent, and dependent-to-head) and the dependency relation between a pair of nodes. Figure 2 displays the dependency graph derived from the dependency tree at the bottom of Figure 1. The self-connection edge is labeled as "self", the head-to-dependent edge is labeled with the original dependency relation (e.g., "nsubj"), and the dependent-to-head edge is labeled with the original dependency relation and ∼ suffix (e.g., "nsubj∼").

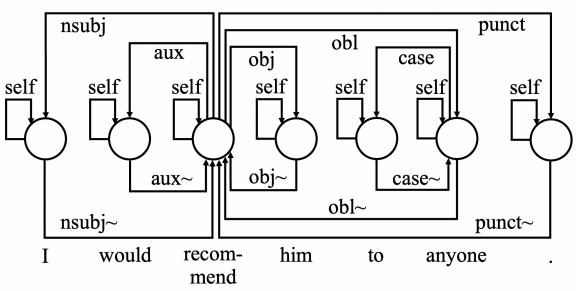
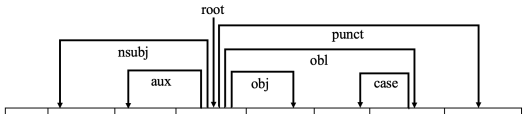


Figure 2: Dependency graph of a sentence converted from its dependency tree.

There are two ways of generating DR to be encoded in the edge representation of the graphs, i.e., A-DR and B-DR. In A-DR, following Marcheggiani and Titov (2017), head-to-dependent edge and dependent-to-head edge that share the same dependency relation, have completely different representations, e.g., "nsubj" (head-to-dependent edge) and "nsubj∼" (dependent-to-head edge) have different representations. In B-DR, we first generate the representations separately for each edge's dependency relation with d_r dimension and dependency direction with d_d dimension. Then, we concatenate both representations to produce the DR with $(d_r + d_d)$ dimension.

In transformer-based models, we encode a sentence as a **fully-connected graph**. Shaw et al. (2018) modify the vanilla Transformers to incorporate relative position representation (RPR) among the edges. We use the same approach to encode the DR among the edges by taking the element-wise addition of DR and the other edge representation if exists, in this case, RPR. Edges in the fully-connected graph that do not have the corresponding edges in the dependency tree, do not have dependency relations. Therefore, we label these edges as "norel" (short for no relation) when constructing DRs.

We encode a sentence in GCN-based and GAT-based models by forming a **dependency graph** based on its dependency tree. We follow the method proposed by Marcheggiani and Titov (2017). They convert a dependency tree to a graph by adding edges that flow in the opposite direction (dependent-to-head) of the original dependency direction (head-to-dependent) and edges that flow from nodes to themselves (self-connection). We encode either A-DR or B-DR in the edge represen-



	I	would	recom-mend	him	to	anyone	.
p_i	0	0	1	0	0	0	0
o_i	PRON	AUX	VERB	PRON	ADP	PRON	PUNCT
a_i	0	1	2	3	4	5	6
s_i	1	1	0	1	2	1	1
d_i	nsubj	aux	root	obj	case	obl	punct

Figure 3: The illustration of predicate indicator, POS tag, absolute position, structural absolute position, and dependency relation of each word in a sentence.

tation.

3.2 Input Layer

To produce the word representation, h_i, for each word in a sentence, we concatenate: (1) predicate indicator embedding (PIE), p_i, which represents whether a word is a predicate or not (Fei et al., 2020), (2) contextualized multilingual word embedding, c_i, which is obtained from the concatenation of the last four hidden layers (Conia et al., 2021) in multilingual BERT (mBERT) (Devlin et al., 2019), (3) POS tag embedding (POSE), o_i, which is the POS tag of each word, (4) absolute position embedding (APE), a_i, which is the position of each word in the sentence (Vaswani et al., 2017), (5) structural absolute position embedding (SAPE), s_i, which is the dependency depth of each word relative to the root of the dependency tree (Wang et al., 2019b), and (6) dependency relation embedding (DRE), d_i, which is the dependency relation attached to each word as a dependent. While embedding (1) and (2) are mandatory, we experiment with embedding (3) - (6) to obtain the optimal combination of embeddings for building the word representation.

Figure 3 shows the example to obtain each embedding. As shown in Equation 1, we apply a dropout to the concatenation result and a linear transformation following Ahmad et al. (2021b).

$$h_i = W(\text{dropout}([p_i, o_i, a_i, s_i, d_i, c_i])) + b \quad (1)$$

3.3 Encoder

We experiment with transformer-based, GCN-based, GAT-based, and BiLSTM-based encoders. We apply an activation function, dropout, and residual connection in consecutive order after each layer for GCN-based and GAT-based encoders. As proposed by GATs (Veličković et al., 2018), we can either apply or drop the activation function at the final layer. For the BiLSTM-based encoder, we only apply the consecutive operations at the final layer.

Transformer-Based Encoders In transformer-based encoders, we experiment with different modifications to vanilla Transformers, i.e., **GATEs** (Ahmad et al., 2021b), self-attention with APE in the node representation (**Transformers**) (Vaswani et al., 2017), self-attention with relative position representation (RPR) in the edge representation (**SAN-RPRs**) (Shaw et al., 2018), self-attention with SAPE in the node representation (**SAN-SAPRs**) (Wang et al., 2019b), and self-attention with structural relative position representation (SRPR) in the edge representation (**SAN-SRPRs**) (Wang et al., 2019b). We also combine SAPE in the node representation with RPR in the edge representation and refer to the self-attention network as **SAPR-RPRs**. Furthermore, we combine APE in the node representation with SRPR in the edge representation and refer to the self-attention network as **Trans-SRPRs**. We measure the dependency distance in SRPR by calculating the number of hops from one node to another based on the dependency tree. Following Ahmad et al. (2019), we take the absolute value when calculating SRPR and RPR to make the models more robust to word order differences. Furthermore, we experiment with incorporating dependency relation representation (DR) (Section 3.1) into the edge representation.

GCN-Based Encoders In GCN-based encoders, we experiment with different types of GCNs, i.e., syntactic GCNs (**SGCNs**) (Marcheggiani and Titov, 2017), relational GCNs (**RGCNs**) (Schlichtkrull et al., 2018), and attention-based relational GCNs (**ARGCNs**) (Jiang et al., 2021). Since SGCNs and RGCNs encode the edge representation by differentiating their weight matrices based on the edge relations, they can only incorporate A-DR (refer to Section 3.1).

GAT-Based Encoders In GAT-based encoders, we experiment with different types of GATs, i.e., **GATs** (Veličković et al., 2018), simple heterogeneous GNNs (**SHGNs**) (Lv et al., 2021), relational GATs (RGATs) (Wang et al., 2020), and knowledge-based GATs (**KBGATs**) (Nathani et al.,

2019). RGATs calculate the second attention weight, β, using position-wise feed-forward network (FFN) (Vaswani et al., 2017). However, we find that the original dot-product equation used to calculate the attention weight proposed by Veličković et al. (2018) works better for this task. Therefore, we modify RGATs into **TAGATs**.

TAGATs calculate the first attention weight, α, using Equation 3 in the original GAT paper (Veličković et al., 2018). To calculate the second attention weight, β, TAGATs slightly modify the equation to incorporate the DR, r_{ij}, as shown in Equation 2, where k is the current attention head, l is the current layer, $\mathcal{N}_i$ is the neighbor nodes of node i, W_r is a weight matrix to linearly transform the DR, r_{ij}, and LR is a Leaky ReLU.

$$\beta_{ij}^{l,k} = \text{softmax}_{j \in \mathcal{N}_i}(\text{LR}(a^{l,k^T}[W_r^{l,k} r_{ij}^l])) \quad (2)$$

Furthermore, TAGATs obtain node representation from attention weight α using Equation 5 and Equation 6 in the original GAT paper (Veličković et al., 2018). To obtain node representation from attention weight β, TAGATs employ Equation 3 and Equation 4. TAGATs concatenate node representations from K heads in the intermediate layers, as shown in Equation 3. Meanwhile, in the final layer, TAGATs take the average of node representations from K heads, as shown in Equation 4, where L is the number of layers.

$$h_{i,\beta}^{l+1} = \sigma(||_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \beta_{ij}^{l,k} W^{l,k} h_j^l), \; l < L \quad (3)$$

$$h_{i,\beta}^{l+1} = \sigma(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \beta_{ij}^{l,k} W^{l,k} h_j^l), \; l = L \quad (4)$$

Finally, TAGATs calculate the node representation for layer $(l+1)$, h_i^{l+1}, by applying a linear transformation to the concatenation of node representation $h_{i,\alpha}^{l+1}$ and node representation $h_{i,\beta}^{l+1}$ and optionally apply an activation function, σ, on top of the linear transformation, as shown in Equation 5.

$$\begin{aligned} x_i^{l+1} &= [h_{i,\alpha}^{l+1}, h_{i,\beta}^{l+1}] \\ h_i^{l+1} &= \sigma(W_{l+1} x_i^{l+1} + b_{l+1}) \end{aligned} \quad (5)$$

3.4 Decoder

We apply a linear scorer as the decoder. For each word, we concatenate sentence representation, h_s, predicate node representation, h_p, and node representation, h_i. h_s is obtained by applying max-pooling over node representations in the sentence (Ahmad et al., 2021b). Meanwhile, h_p is taken from the node representation of the sentence's predicate. After that, following Ahmad et al. (2021b), we apply two feed-forward neural networks (FFNNs), each followed by a ReLU (RL), to produce the final node representation, h_f, with the dimension equal to the number of arguments, c, as shown in Equation 6.

$$h_f = \text{RL}(W_2(\text{RL}(W_1[h_s, h_p, h_i] + b_1)) + b_2) \quad (6)$$

Finally, we apply a softmax function to produce the probability for each argument, z, as shown in Equation 7. We train the model to minimize the cross-entropy loss.

$$P(z) = \text{softmax}_z(h_f), \; z \in [1, c] \quad (7)$$

4 Experiments

4.1 Corpus

We conduct experiments using the corpus from UPB v2[4] (Jindal et al., 2022). UPB v2 contains SRL annotations based on dependency tree annotations in UD v2.9. Appendix A.2 shows some preprocessing steps that we run to fix the shifted annotations in languages with enhanced dependency tree annotations. In our experiments, we merge all treebanks that belong to the same language.

4.2 Settings

We focus on conducting experiments in a zero-shot setting, training the model in English and evaluating the model in 23 target languages provided by UPB v2, as shown in Table 1.

In the training phase, we take the model from the best epoch based on the evaluation using the English validation set. After that, we find each model's best configuration based on the average F1 scores of the validation sets in 23 languages. Finally, we compare the transferability of different models using the same strategy. The reason of using the validation sets in 23 languages is that we want to focus on how well each model can

[4]https://github.com/UniversalPropositions

Model	Description
GATEs	graph attention transformer encoders
Transformers	self-attention with absolute position embedding (APE) in node
SAN-RPRs	self-attention with relative position representation (RPR) in edge
SAN-SAPRs	self-attention with structural absolute position embedding (SAPE) in node
SAN-SRPRs	self-attention with structural relative position representation (SRPR) in edge
SAPR-RPRs	combination of SAN-SAPRs and SAN-RPRs
Trans-SRPRs	combination of Transformers and SAN-SRPRs
SAPR-RPR-DRs	SAPR-RPRs with dependency relation representation (DR) in edge
Trans-SRPR-DRs	Trans-SRPRs with DR in edge
SGCNs	syntactic graph convolutional networks (GCNs)
RGCNs	relational GCNs
ARGCNs	attention-based relational GCNs
GATs	graph attention networks
SHGNs	simple heterogeneous GNNs
KBGATs	knowledge-based GATs
TAGATs	two-attention relational GATs
BiLSTMs	bidirectional LSTMs

Table 2: Summary of models involved in the experiments.

Model	Node Representation			Edge Representation		
	APE	SAPE	DRE	RPR	DR	SRPR
GATEs		✓	✓			
Transformers	✓		✓			
SAN-RPRs			✓	✓		
SAN-SAPRs		✓				
SAN-SRPRs						✓
SAPR-RPRs		✓		✓		
Trans-SRPRs	✓		✓			✓
SAPR-RPR-DRs		✓			B	
Trans-SRPR-DRs	✓		✓		B	✓
SGCNs			✓		A	
RGCNs		✓			A	
ARGCNs			✓	✓	A	
GATs		✓	✓			
SHGNs					A	
KBGATs			✓	✓	B	
TAGATs		✓		✓	A	
BiLSTMs		✓	✓			

Table 3: Best combination in node and edge representations of each model. SAPE, DRE, DR, and SRPR are feature representations derived from the dependency tree.

generalize to other languages. Choosing the best configuration for each model based solely on the English validation set may result in a model that focuses too heavily on features specific to English.

We run comprehensive experiments to find the best configuration for each model including feature combination in node representation, feature combination in edge representation, number of layers, dropouts, and activation functions (Appendix B.2). Table 2 summarizes the models we compare in the experiments. Table 3 shows each model's best feature combination in node and edge representations. We can see that at least one feature representation from the dependency tree is required to produce the best configuration for each model.

We use predicted dependency trees and POS tags for model evaluation (Appendix B.1) obtained from pre-trained models trained on UD 2.8 (Qi et al., 2020). We use frozen mBERT as our contextualized word embedding to observe each model's effectiveness solely, eliminating fine-tuned mBERT as one of the factors that might affect the model performance. Our preliminary experiments show that taking the average of subword embeddings from mBERT works best for our task, so we apply this setting to all of our experiments. We report the average F1 scores from five runs with the standard deviation for model comparison.

4.3 Comparison Among Transformer-Based Models

Table 4 compares transformer-based models with the best configuration. Trans-SRPR-DRs and SAPR-RPR-DRs perform worst among the transformer-based models. To obtain DR in the fully-connected graph, we label the edges that do not reflect the edges in the corresponding dependency tree as "norel". However, naturally, many more edges are labeled as "norel" than edges labeled as the actual dependency relations.[5] The imbalanced proportion of dependency relations (i.e., number of edges labeled with "norel" is many more than number of edges labeled with actual dependency relations) might cause the model to overfit.

Furthermore, the table shows the superiority of SAN-RPRs over Transformers, indicating that encoding the position of each word relative to another word in the edge representation produces a more general model than encoding the absolute position of each word in the sentence in the node representation. This finding aligns with the results in Ahmad et al. (2019). On the other hand, SAN-SAPRs outperform SAN-SRPRs, indicating that encoding each word's dependency distance relative to the root of the dependency tree in the node representation produces a more general model than encoding each word's dependency distance relative to another word in the edge representation. We further combine the information regarding the position of each word according to the sentence and the sentence's dependency tree, i.e., SAPR-RPRs and

[5]Each word can only have one parent in the dependency tree. Given that n indicates the number of nodes, each dependency graph will have $(n-1)$ edges labeled with dependency relations, $(n-1)$ edges labeled with opposite dependency relations, and n edges that connect each node with itself. However, since there are n^2 edges in the fully-connected graph, there will be (n^2-3n+2) edges labeled as "norel".

Model	PR	EN	AVG
GATEs	12.2M	78.96±0.31	52.57±0.23
Transformers	12.2M	76.16±0.51	52.07±0.21
SAN-RPRs	12.2M	78.26±0.40	**52.73±0.40**
SAN-SAPRs	12.2M	75.93±0.47	51.98±0.10
SAN-SRPRs	12.1M	78.27±0.50	51.51±0.27
SAPR-RPRs	12.2M	78.11±0.42	52.64±0.38
Trans-SRPRs	12.2M	79.03±0.32	52.21±0.30
SAPR-RPR-DRs	12.2M	79.83±0.19	50.69±0.21
Trans-SRPR-DRs	12.2M	79.85±0.21	50.60±0.14
SGCNs	5.99M	79.94±0.27	**52.52±0.38**
RGCNs	3.23M	78.28±0.29	51.48±0.35
ARGCNs	3.52M	77.84±0.44	52.13±0.32
GATs	5.09M	79.81±0.19	52.66±0.14
SHGNs	5.06M	78.84±0.35	52.61±0.26
KBGATs	7.73M	79.53±0.31	52.31±0.32
TAGATs	6.31M	79.07±0.19	**52.78±0.14**

Table 4: F1 scores (%) of transformer-based, GCN-based, and GAT-based models evaluated on UPB v2 test set. AVG indicates the average F1 scores of a specific model evaluated in target languages. The bold and underlined scores indicate each group's highest and second-highest scores. PR is the number of parameters in each model.

Trans-SRPRs. Consistent with the previous results, SAPR-RPRs, which consist of features from SAN-SAPRs and SAN-RPRs, outperform Trans-SRPRs, which consist of features from Transformers and SAN-SRPRs.

According to the average F1 score, SAN-RPRs and SAPR-RPRs are both strong models with 52.73% and 52.64% average F1 scores, respectively, outperforming GATEs (Ahmad et al., 2021b), i.e., the model proposed for cross-lingual EARL. Although SAN-RPRs have a better average F1 score than SAPR-RPRs, based on our observation among transformer-based models, SAPR-RPRs show their superiority in more languages (Appendix C.1.1). Therefore, we compare both models with other best models in Section 4.6.

4.4 Comparison Among GCN-Based Models

Table 4 compares GCN-based models with the best configuration. SGCNs significantly outperform the other GCN-based models, i.e., RGCNs and ARGCNs, with a 52.52% average F1 score. RGCNs perform the worst among GCN-based models because RGCNs are the only networks that do not apply the attention mechanism. ARGCNs apply a self-attention mechanism, while SGCNs implement the attention mechanism as a gating mechanism. Self-attention or gating mechanism measures how much attention each node should pay to other nodes when the network updates each node's representation. Those mechanisms help to emphasize the edges in the dependency graph that co-occur with the predicate-argument paths.

4.5 Comparison Among GAT-Based Models

Table 4 compares GAT-based models with the best configuration. TAGATs perform better than SHGNs, indicating that the GAT-based model learns better when attention weight calculation is separated based on node representations and edge representations. Moreover, GATs also perform better than SHGNs indicating that encoding the SAPE and DRE in the node representation is a better way to encode dependency features than combining node representations with edge representations when calculating the attention weight. According to the average F1 score, TAGATs and GATs are both strong models with 52.78% and 52.66% average F1 scores, respectively. Therefore, we compare both models with other best models in Section 4.6.

On the other hand, according to the average F1 score, KBGATs perform worst among GAT-based models. The significant difference between KBGATs and the other GAT-based models is that KBGATs update each node representation with (1) neighbor node representations and (2) surrounding edges' representations that contain information about DR and RPR. Meanwhile, the other models update each node representation only with (1). We conjecture that the approach of KBGATs might overpopulate each node in every update with information too specific to a particular language the network learns from.

4.6 Comparison Among Best Models

We compare the best models from each group, i.e., SAN-RPRs, SAPR-RPRs, SGCNs, GATs, and TAGATs, with the BiLSTM-based model, i.e., BiLSTMs, in Table 5. We calculate each model's superiority score (SC) based on the model performance in target languages. We allocate 2 points if the model achieves the highest F1 score or 1 point if the model achieves the second-highest F1 score for a specific language.

TAGATs have the best average F1 score among the models, with a 52.78% average F1 score. However, SAPR-RPRs and GATs perform best among the models in slightly more languages, as indicated by the higher SCs. Despite having the second-best average F1 score of 52.73%, SAN-RPRs have a lower SC than SAPR-RPRs, GATs, and TAGATs. Overall, transformer-based and GAT-based models outperform SGCNs, indicating that the self-attention mechanism better emphasizes the essen-

	d ≥ 3	SAN-RPRs	SAPR-RPRs	SGCNs	GATs	TAGATs	BiLSTMs
EN	2.68	78.26±0.40	78.11±0.42	79.94±0.27	79.81±0.19	79.07±0.19	76.86±0.34
AVG	-	52.73±0.40	52.64±0.38	52.52±0.38	52.66±0.14	**52.78±0.14**	51.85±0.09
TA	17.18	37.96±1.75	**39.57±1.18**	34.32±1.12	35.08±0.58	35.68±1.28	34.19±1.22
HI	8.46	47.51±0.62	45.04±0.35	48.24±0.61	**48.25±0.33**	47.65±0.33	46.63±0.38
ZH	8.41	50.37±1.17	**50.96±0.88**	45.77±0.67	46.12±0.39	46.80±0.64	47.56±0.89
JA	8.11	37.69±0.99	34.78±1.29	37.43±0.28	37.99±0.52	**39.30±0.73**	37.40±0.61
VI	7.89	28.69±0.79	**29.10±0.45**	27.95±0.56	28.06±0.59	28.31±0.55	28.18±0.88
KO	6.88	42.61±1.84	**45.24±1.23**	42.92±0.64	43.22±0.56	44.57±0.24	41.77±1.61
ID	5.52	58.78±1.09	**59.97±0.53**	58.54±0.82	58.33±0.69	59.11±0.87	56.11±0.84
HU	5.38	49.76±0.35	49.08±0.34	50.76±0.41	**51.10±0.51**	50.90±0.37	50.64±0.39
RO	5.32	54.23±0.67	**54.46±0.52**	53.57±0.47	54.12±0.49	53.60±0.45	53.26±0.34
FR	4.55	**62.19±0.41**	62.11±0.47	60.93±0.38	61.64±0.44	61.13±0.22	61.22±0.27
MR	4.08	**41.06±2.89**	40.36±2.20	40.97±3.40	38.06±0.13	39.26±1.20	37.18±2.28
UK	4.06	58.92±0.26	59.36±0.72	59.66±0.76	59.49±0.56	**59.72±0.31**	58.96±0.07
PT	3.75	66.05±0.21	**66.49±0.33**	65.62±0.43	65.99±0.32	65.61±0.15	64.40±0.33
IT	3.73	**58.11±0.33**	57.80±0.39	57.43±0.42	58.00±0.42	57.34±0.34	58.02±0.27
ES	3.67	63.71±0.33	63.62±0.25	63.87±0.61	**64.29±0.36**	63.91±0.27	62.48±0.29
CS	3.66	56.87±0.27	55.80±0.51	57.95±0.52	**58.02±0.21**	57.62±0.28	56.59±0.36
EL	3.59	60.59±0.23	60.23±0.40	60.56±0.69	60.74±0.48	**60.86±0.34**	59.76±0.45
FI	3.35	**55.58±0.42**	55.29±0.54	54.87±0.40	54.62±0.32	54.88±0.20	54.62±0.20
RU	3.07	60.18±0.44	**61.13±0.50**	59.98±0.34	60.14±0.16	60.30±0.22	59.73±0.25
NL	3.05	62.84±0.37	62.22±0.58	62.94±0.21	**63.53±0.64**	62.97±0.37	62.47±0.36
TE	2.49	44.66±2.00	43.88±1.57	46.08±1.07	**46.96±1.49**	**46.96±1.82**	45.95±0.51
DE	2.46	56.86±0.32	56.98±1.13	**58.61±0.30**	58.52±0.26	58.52±0.18	57.72±0.23
PL	1.71	57.67±0.36	57.28±0.46	**59.08±0.31**	59.00±0.44	58.92±0.52	57.73±0.23
SC	-	13	18	9	16	15	1
PR	-	12.2M	12.2M	5.99M	5.09M	6.31M	9.03M

Table 5: F1 scores (%) of best models evaluated on UPB v2 test set with predicted parsers. The bold score and underlined score indicate the highest and second-highest scores. AVG indicates the average F1 scores of a specific model evaluated in target languages. PR and SC are the number of parameters and the superiority score of each model. d≥3 column indicates the proportion of gold arguments (%) that fall in ≥3 dependency distance. We use predicted dependency trees to measure d.

tial dependency paths than the gating mechanism. Finally, BiLSTMs perform the worst as the only network that encodes sentences sequentially.

According to our experiments, stacking two layers of GAT-based encoders performs best. However, this approach has a drawback, as the information can only travel as far as two hops from the origin node. On the other hand, stacking three layers of transformer-based encoders does not affect the information's traveling distance. In transformer-based models, we construct a fully-connected graph of a sentence, allowing information to travel from one node to every other node, regardless of how many layers are stacked together.

The second column in Table 5 shows the percentage of arguments in ≥3 dependency distance, i.e., the number of hops from the predicate node to a specific node according to the sentence's dependency tree. Some languages like Tamil (TA), Hindi (HI), Chinese (ZH), Japanese (JA), and Vietnamese (VI) have a relatively high number of arguments (>7%) in ≥3 dependency distance. Transformer-based models perform significantly better in TA (>3%) and ZH (>4%), slightly better in VI (<1%), and slightly worse in HI (<1%) and JA (<2%). This evidence proves that transformer-based models are generally better in long-range dependency

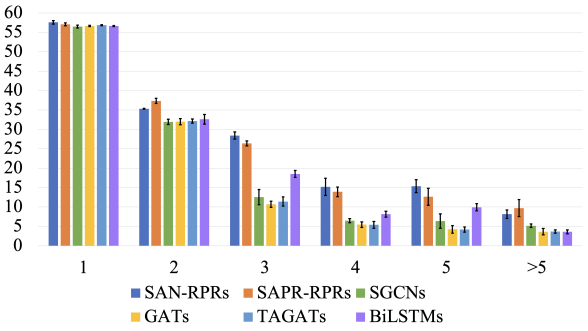


Figure 4: Average F1 scores (%) of best models grouped by the dependency distance evaluated on UPB v2 test set with predicted parsers.

distance than GAT-based models. On the other hand, some languages have a relatively low number of arguments (<3%) in ≥3 dependency distance, i.e., Telugu (TE), German (DE), and Polish (PL). For these languages, where most of the arguments lie in 1-2 dependency distance, GAT-based models perform better than transformer-based models.

Figure 4 shows the average F1 scores of each model grouped by the dependency distance. Although BiLSTMs perform worst among the models, BiLSTMs are better in long-range dependency distance than GCN-based and GAT-based models. The figure also shows the superiority of

Model	SAN-RPRs	SAPR-RPRs
base	52.73±0.40	52.64±0.38
w/o POSE	52.73±0.32	51.91±0.46
w/o PIE	51.54±0.36	51.14±0.43
w/o SAPE	-	51.65±0.28
w/o DRE	51.65±0.28	-

Table 6: Average F1 scores (%) of SAN-RPRs and SAPR-RPRs with fine-tuned mBERT and certain embedding removed from the node evaluated on UPB v2 test set with predicted parsers.

transformer-based models among all models as the dependency distance increases.

4.7 Ablation Study

We conduct ablation studies for the best transformer-based models, i.e., SAN-RPRs and SAPR-RPRs, as shown in Table 6. We experiment with removing DRE, POSE, or PIE from the node representation in SAN-RPRs. Removing either DRE or PIE from the node representation reduces the performance of SAN-RPRs. However, based on our observation, SAN-RPRs without POSE transfer better to most languages (Appendix C.1.2).

Furthermore, we also experiment with removing SAPE, POSE, or PIE from the node representation in SAPR-RPRs. Removing SAPE, POSE, or PIE from the node representation reduces the performance of SAPR-RPRs. We conjecture that the combination of POSE and SAPE to provide syntactic information in SAPR-RPRs is necessary to replace the role of DRE in SAN-RPRs.

In Table 5, based on SC, we can see that SAPR-RPRs perform best in more languages than SAN-RPRs, even though the average F1 score of SAPR-RPRs is worse than SAN-RPRs. Since our experiments conclude that SAN-RPRs without POSE perform better in most languages, we re-compare SAN-RPRs without POSE with the other best models taken from Table 5. Based on our observation (Appendix C.2.1), after removing POSE from SAN-RPRs, the model performs best in more languages than SAPR-RPRs.

In Table 7, we show the comparison of SAN-RPRs without POSE with predicted parsers, i.e., SAN-RPRs (pred), and gold parsers, i.e., SAN-RPRs (gold), along with the unlabeled attachment score (UAS) and labeled attachment score (LAS) of the predicted parsers. For languages with LAS<80%, i.e., TA (72.30%), Indonesian (ID) (77.33%), and Marathi (MR) (70.63%), the F1 score difference is ±2%. However, in most languages where LAS>80%, the F1 score difference is less than 1% indicating that the inaccuracy of predicted parsers is relatively negligible.

Lang	SAN-RPRs (pred)	SAN-RPRs (gold)	UAS	LAS
EN	78.32±0.32	79.73±0.35	91.42	89.82
AVG	52.73±0.32	53.01±0.33	-	-
TA	36.53±1.32	38.56±1.42	80.89	72.30
HI	47.69±1.10	47.46±0.95	96.68	94.43
ZH	50.81±0.51	50.78±0.38	87.06	85.13
JA	36.07±1.20	35.82±1.27	95.21	94.61
VI	28.30±0.92	28.30±0.85	77.58	74.16
KO	42.06±1.50	41.74±1.50	90.19	88.76
ID	57.77±0.83	60.33±0.65	87.31	77.33
HU	50.02±0.42	50.78±0.61	86.72	83.25
RO	54.94±0.50	54.97±0.53	92.25	89.12
FR	62.36±0.59	62.79±0.54	90.69	88.04
MR	41.83±3.50	43.50±3.19	79.85	70.63
UK	58.84±0.59	58.53±0.56	90.10	88.24
PT	66.65±0.44	66.80±0.36	94.51	93.38
IT	58.65±0.49	58.70±0.61	91.09	88.43
ES	64.07±0.51	64.34±0.51	93.47	91.55
CS	57.20±0.14	57.11±0.15	93.48	91.81
EL	60.86±0.56	60.75±0.68	92.93	91.19
FI	55.40±0.16	55.45±0.15	93.02	91.47
RU	60.12±0.36	60.74±0.35	87.58	84.46
NL	63.21±0.38	63.14±0.44	92.49	89.75
TE	43.46±2.02	42.63±2.36	93.07	85.58
DE	57.70±0.43	57.94±0.40	95.04	93.34
PL	58.16±0.35	58.10±0.37	95.48	94.21

Table 7: The left-hand side shows the comparison of F1 scores (%) of SAN-RPRs w/o POSE evaluated with predicted dependency parsers and gold dependency parsers on the UPB v2 test set. The right-hand side shows UAS (%) and LAS (%) of predicted dependency parsers.

5 Conclusions and Future Work

Through comprehensive experiments, we consistently show that incorporating syntax from dependency trees can improve the transferability of cross-lingual SRL models across languages. Overall, we show that the transformer-based model, i.e., SAN-RPRs that encode DRE without POSE in the node representation and RPR in the edge representation, stacked in three layers, is the most transferable among all models, especially as the dependency distance increases. However, TAGATs that encode SAPE in the node representation and DR and RPR in the edge representation, stacked in two layers, transfer better than SAN-RPRs in languages where most of the arguments lie in 1-2 dependency distance.

In the future, we can extend the model training to a few-shot setting where we include a certain proportion of target sentences in the training set. We can modify the network's objective function and/or parameters to maximize the learning of universal features without ignoring the language-specific features.

6 Limitations

This work's limitation is that we focus on argument detection and argument labeling in cross-lingual SRL, assuming that the sentences' gold predicates are easy to obtain. Furthermore, we focus on conducting experiments in a zero-shot setting. The availability of target sentences in the training set might affect the models' behavior, which should be investigated further.

7 Ethics Statement

We believe there is no ethical issue raised in this work. SRL is a low-level task that supports other advanced NLP applications. Therefore, increasing the coverage of SRL models in various languages is beneficial for developing NLP tools that can help solve problems in this diverse society.

References

Wasi Ahmad, Haoran Li, Kai-Wei Chang, and Yashar Mehdad. 2021a. Syntax-augmented multilingual BERT for cross-lingual transfer. 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 4538–4554, Online. Association for Computational Linguistics.

Wasi Ahmad, Nanyun Peng, and Kai-Wei Chang. 2021b. Gate: Graph attention transformer encoder for cross-lingual relation and event extraction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(14):12462–12470.

Wasi Ahmad, Zhisong Zhang, Xuezhe Ma, Eduard Hovy, Kai-Wei Chang, and Nanyun Peng. 2019. On difficulties of cross-lingual transfer with order differences: A case study on dependency parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2440–2452, Minneapolis, Minnesota. Association for Computational Linguistics.

Alan Akbik, Laura Chiticariu, Marina Danilevsky, Yunyao Li, Shivakumar Vaithyanathan, and Huaiyu Zhu. 2015. Generating high quality proposition Banks for multilingual semantic role labeling. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 397–407, Beijing, China. Association for Computational Linguistics.

Rui Cai and Mirella Lapata. 2020. Alignment-free cross-lingual semantic role labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3883–3894, Online. Association for Computational Linguistics.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2010. Semantic role labeling for open information extraction. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 52–60, Los Angeles, California. Association for Computational Linguistics.

Simone Conia, Andrea Bacciu, and Roberto Navigli. 2021. Unifying cross-lingual semantic role labeling with heterogeneous linguistic resources. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 338–351, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Hao Fei, Meishan Zhang, and Donghong Ji. 2020. Cross-lingual semantic role labeling with high-quality translated training corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7014–7026, Online. Association for Computational Linguistics.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1243–1252. JMLR.org.

Aleksa Gordić. 2020. pytorch-gat. `https://github.com/gordicaleksa/pytorch-GAT`.

Junfeng Jiang, An Wang, and Akiko Aizawa. 2021. Attention-based relational graph convolutional network for target-oriented opinion words extraction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1986–1997, Online. Association for Computational Linguistics.

Ishan Jindal, Alexandre Rademaker, Michał Ulewicz, Ha Linh, Huyen Nguyen, Khoi-Nguyen Tran, Huaiyu Zhu, and Yunyao Li. 2022. Universal Proposition Bank 2.0. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 1700–1711, Marseille, France. European Language Resources Association.

Atif Khan, Naomie Salim, and Yogan Jaya Kumar. 2015. A framework for multi-document abstractive summarization based on semantic role labelling. *Applied Soft Computing*, 30:737–747.

Jack Kiefer and Jacob Wolfowitz. 1952. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, pages 462–466.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. 2021. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '21, page 1150–1160, New York, NY, USA. Association for Computing Machinery.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark. Association for Computational Linguistics.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4710–4723, Florence, Italy. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association (ELRA).

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Reinhard Rapp. 2022. Using semantic role labeling to improve neural machine translation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 3079–3083, Marseille, France. European Language Resources Association.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Gabriel Stanovsky and Ido Dagan. 2016. Creating a large benchmark for open information extraction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2300–2305, Austin, Texas. Association for Computational Linguistics.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England. Coling 2008 Organizing Committee.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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

Kai Wang, Weizhou Shen, Yunyi Yang, Xiaojun Quan, and Rui Wang. 2020. Relational graph attention network for aspect-based sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3229–3238, Online. Association for Computational Linguistics.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019a. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*.

Xing Wang, Zhaopeng Tu, Longyue Wang, and Shuming Shi. 2019b. Self-attention with structural position representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1403–1409, Hong Kong, China. Association for Computational Linguistics.

Zhisong Zhang, Emma Strubell, and Eduard Hovy. 2021. On the benefit of syntactic supervision for cross-lingual transfer in semantic role labeling. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6229–6246, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Lang	Train	Dev	Test
English (EN)	12,542	1,974	2,062
Chinese (ZH)	3,997	500	500
Czech (CS)	102,993	11,311	12,203
Dutch (NL)	18,078	1,394	1,472
Finnish (FI)	27,198	3,239	3,422
French (FR)	17,968	2,970	1,712
German (DE)	166,849	19,233	19,436
Greek (EL)	1,662	403	456
Hindi (HI)	13,304	1,659	1,684
Hungarian (HU)	910	441	449
Indonesian (ID)	4,482	559	557
Italian (IT)	29,685	2,277	2,518
Japanese (JA)	14,100	1,014	1,086
Korean (KO)	27,410	3,016	3,276
Marathi (MR)	373	46	47
Polish (PL)	31,496	3,960	3,942
Portuguese (PT)	16,633	2,376	2,367
Romanian (RO)	35,911	2,247	2,272
Russian (RU)	19,894	1,525	1,482
Spanish (ES)	28,474	3,054	2,147
Tamil (TA)	400	80	120
Telugu (TE)	1,051	131	146
Ukrainian (UK)	5,496	672	892
Vietnamese (VI)	1,400	800	800

Table 8: Number of sentences available in each language in UPB v2.

A Artifacts

A.1 Corpus Distribution

Table 8 shows the corpus distribution in UPB v2. Since we run our experiments in a zero-shot setting, we only use the dev set and test set for languages other than English.

A.2 Corpus Preprocessing

Some treebanks have enhanced dependency tree annotations that cause new tokens to be added to the sentences. These tokens are called enhanced tokens. The enhanced tokens cause some SRL annotations in UPB v2 to be shifted when merged with UD v2.9, resulting in the wrong predicate or semantic role annotations. For example, take a look at the example of wrong predicate annotation taken from the dev set in `Finnish-TDT` (UPB v2). Token 10.1 is an enhanced token.

```
# sent_id = w063.9
# text = Osasto.........
1 _ _ _
2 _ _ _
3 be.01 A1:2|AM-LOC:7 A1:1-2|AM-LOC:6-7
4 _ _ _
5 _ _ _
6 _ _ _
7 _ _ _
8 _ _ _
9 _ _ _
10 _ _ _
10.1 _ _ _
11 be.01 A1:9 A1:9-10
12 _ _ _
13 _ _ _
```

The corresponding annotation in UD v2.9 is as follows. Note that we only present the tokenized words, lemmas, and POS tags here since we only show the UD annotation to highlight the shifted annotation problem.

```
# sent_id = w063.9
# text = Osasto N7 sijaitsee
  samassa korttelissa
  Naistenklinikan rakennuksessa
  ja osasto LV37 Kätilöopiston
  sairaalassa.
1 Osasto osasto NOUN
2 N7 N7 SYM
3 sijaitsee sijaita VERB
4 samassa sama PRON
5 korttelissa kortteli NOUN
6 Naistenklinikan nais#klinikka
  NOUN
7 rakennuksessa rakennus NOUN
8 ja ja CCONJ
9 osasto osasto NOUN
10 LV37 LV37 SYM
10.1 sijaitsee sijaita VERB
11 Kätilöopiston kätilö#opisto
   NOUN
12 sairaalassa sairaala NOUN
13 . . PUNCT
```

If we compare the two annotations from UPB v2 and UD v2.9, we can see that the first predicate annotated on token 3, i.e., “sijaitsee”, is correct. However, the second predicate annotated on the token 11, i.e., “Kätilöopiston”, is wrong. The correct second predicate is token 10.1, i.e., “sijaitsee”. The annotation is somehow shifted because of the enhanced token added, token 10.1. Therefore, we fix the annotation in UPB v2 to be as follows.

```
# sent_id = w063.9
# text = Osasto.........
1 _ _ _
2 _ _ _
3 be.01 A1:2|AM-LOC:7 A1:1-2|AM-LOC:6-7
4 _ _ _
5 _ _ _
6 _ _ _
7 _ _ _
```

```
8 _ _ _
9 _ _ _
10 _ _ _
10.1 be.01 A1:9 A1:9-10
11 _ _ _
12 _ _ _
13 _ _ _
```

In some cases, the predicate annotations and the semantic role annotations are shifted. We run a script to fix the annotation problems in all treebanks with enhanced dependency tree annotations. The treebanks with enhanced dependency tree annotations are `Czech-CAC`, `Czech-FicTree`, `Czech-PDT`, `Dutch-Alpino`, `Dutch-LassySmall`, `Finnish-TDT`, `Italian-ISDT`, `Spanish-AnCora`, and `Ukrainian-IU`.

After we fix the shifted predicate and semantic role annotations, we notice that some predicates and their semantic roles are annotated in the enhanced tokens that do not appear in the original sentence. The treebanks that contain this phenomenon are `Dutch-Alpino`, `Dutch-LassySmall`, `Finnish-TDT`, `Ukrainian-IU`, and `Spanish-AnCora`. We cannot accommodate these annotations since we build the models based on the sentence's original tokens. Therefore, we omit the predicates and their corresponding semantic roles annotated on enhanced tokens in our experiments.

A.3 License

Complete UPB v2 contains annotations from UD v2.9. Table 9 shows the license for each treebank in UD v2.9. In addition to licenses inherited from UD v2.9, UPB v2 also has a CDLA-Sharing-1.0 license.

We refer to publicly available codes to build the corpus and models for experiments. We provide the following list of GitHub repositories with their corresponding licenses.

1. UniversalPropositions/tools: Apache-2.0
2. diegma/neural-dep-srl (Marcheggiani and Titov, 2017): Apache-2.0
3. AnWang-AI/towe-eacl (Jiang et al., 2021): No License
4. dmlc/dgl (Wang et al., 2019a): Apache-2.0
5. gordicaleksa/pytorch-GAT (Gordić, 2020): MIT
6. deepakn97/relationPrediction (Nathani et al., 2019): No License
7. thudm/hgb (Lv et al., 2021): No License
8. shenwzh3/RGAT-ABSA (Wang et al., 2020): MIT
9. wasiahmad/GATE (Ahmad et al., 2021b): MIT

Treebank	License
English-EWT	CC BY-SA 4.0
Chinese-GSD	CC BY-SA 4.0
Czech-CAC	CC BY-SA 4.0
Czech-CLTT	CC BY-SA 4.0
Czech-FicTree	CC BY-NC-SA 4.0
Czech-PDT	CC BY-NC-SA 3.0
Dutch-Alpino	CC BY-SA 4.0
Dutch-LassySmall	CC BY-SA 4.0
Finnish-FTB	CC BY 4.0
Finnish-TDT	CC BY-SA 4.0
French-GSD	CC BY-SA 4.0
French-Rhapsodie	CC BY-SA 4.0
French-Sequoia	LGPL-LR
German-GSD	CC BY-SA 4.0
German-HDT	CC BY-SA 4.0
Greek-GDT	CC BY-NC-SA 3.0
Hindi-HDTB	CC BY-NC-SA 4.0
Hungarian-Szeged	CC BY-NC-SA 3.0
Indonesian-GSD	CC BY-SA 4.0
Italian-ISDT	CC BY-NC-SA 3.0
Italian-ParTUT	CC BY-NC-SA 4.0
Italian-PoSTWITA	CC BY-NC-SA 4.0
Italian-TWITTIRO	CC BY-SA 4.0
Italian-VIT	CC BY-NC-SA 3.0
Japanese-GSD	CC BY-SA 4.0
Japanese-GSDLUW	CC BY-SA 4.0
Korean-GSD	CC BY-SA 4.0
Korean-Kaist	CC BY-SA 4.0
Marathi-UFAL	CC BY-SA 4.0
Polish-LFG	GNU GPL 3.0
Polish-PDB	CC BY-NC-SA 4.0
Portuguese-Bosque	CC BY-SA 4.0
Portuguese-GSD	CC BY-SA 4.0
Romanian-Nonstandard	CC BY-SA 4.0
Romanian-RRT	CC BY-SA 4.0
Romanian-SiMoNERo	CC BY-SA 4.0
Russian-GSD	CC BY-SA 4.0
Russian-Taiga	CC BY-SA 4.0
Spanish-AnCora	CC BY 4.0
Spanish-GSD	CC BY-SA 4.0
Tamil-TTB	CC BY-NC-SA 3.0
Telugu-MTG	CC BY-SA 4.0
Ukrainian-IU	CC BY-NC-SA 4.0
Vietnamese-VTB	CC BY-SA 4.0

Table 9: License for each treebank in UD v2.9.

We access all the resources mentioned above solely for academic research. We follow the intended usage of each artifact.

Treebank	Dev			Test		
	$\mathbf{F1_{POS}}$	UAS	LAS	$\mathbf{F1_{POS}}$	UAS	LAS
English-EWT	96.79	92.46	90.86	96.80	91.42	89.82
Chinese-GSD	95.35	85.11	83.19	95.52	87.06	85.13
Czech-CAC	99.26	92.97	91.62	98.70	93.43	91.68
Czech-CLTT	99.44	89.13	86.98	98.98	88.32	86.09
Czech-FicTree	98.43	94.68	93.11	98.34	94.61	92.76
Czech-PDT	98.77	93.74	92.24	98.63	93.50	91.87
Dutch-Alpino	98.36	94.53	92.24	97.33	92.87	90.42
Dutch-LassySmall	97.03	90.77	87.62	96.31	92.12	89.11
Finnish-FTB	96.90	93.77	92.29	96.87	94.03	92.41
Finnish-TDT	98.08	91.97	90.41	97.78	92.24	90.74
French-GSD	98.45	95.66	94.45	98.20	93.47	91.87
French-Rhapsodie	98.12	87.75	83.25	97.64	86.42	81.88
French-Sequoia	99.03	93.54	92.23	99.12	93.10	91.70
German-GSD	96.19	91.78	88.61	95.37	89.65	85.62
German-HDT	98.08	95.18	93.64	98.30	95.30	93.72
Greek-GDT	97.74	91.77	90.43	97.71	92.93	91.19
Hindi-HDTB	97.89	96.62	94.49	97.93	96.68	94.43
Hungarian-Szeged	96.66	87.64	84.10	96.06	86.72	83.25
Indonesian-GSD	94.64	86.49	76.25	94.73	87.31	77.33
Italian-ISDT	98.54	94.41	92.84	98.62	94.37	93.16
Italian-ParTUT	97.86	92.76	90.52	98.54	93.10	91.40
Italian-PoSTWITA	97.35	87.21	83.20	96.96	88.33	84.41
Italian-TWITTIRO	96.79	87.25	81.64	96.20	84.85	79.77
Italian-VIT	98.12	90.63	88.82	98.16	91.54	89.05
Japanese-GSD	98.34	96.09	95.47	98.10	95.11	94.21
Japanese-GSDLUW	98.54	96.12	95.82	98.58	95.35	95.12
Korean-GSD	95.79	88.22	85.41	96.27	89.65	87.07
Korean-Kaist	96.19	91.35	90.39	95.58	90.41	89.45
Marathi-UFAL	89.32	74.55	64.32	90.53	79.85	70.63
Polish-LFG	98.94	97.56	96.73	99.05	97.80	96.92
Polish-PDB	98.75	94.17	92.69	98.74	94.58	93.16
Portuguese-Bosque	97.92	94.25	92.51	98.10	94.85	93.54
Portuguese-GSD	98.36	94.44	93.34	98.28	94.21	93.23
Romanian-Nonstandard	96.77	93.18	90.04	96.40	91.43	87.75
Romanian-RRT	98.06	91.96	88.60	97.92	91.93	88.45
Romanian-SiMoNERo	98.19	93.38	91.21	98.23	93.78	91.86
Russian-GSD	98.38	90.55	87.80	98.09	90.44	87.21
Russian-Taiga	95.80	83.94	79.32	97.06	84.42	81.41
Spanish-AnCora	98.99	93.83	92.16	98.96	93.82	92.00
Spanish-GSD	97.13	91.91	89.79	97.26	91.93	89.58
Tamil-TTB	87.17	81.24	73.48	86.93	80.89	72.30
Telugu-MTG	94.41	92.90	86.25	94.45	93.07	85.58
Ukrainian-IU	98.08	91.14	89.34	97.67	90.10	88.24
Vietnamese-VTB	92.84	78.92	74.99	92.81	77.58	74.16

Table 10: $\mathrm{F1_{POS}}$, UAS, and LAS of each treebank's POS tagger and dependency parser in UPB v2. $\mathrm{F1_{POS}}$ indicates the F1 score of the POS tagger.

Hyperparameter	Value
num_epochs	100
batch_size	32
optimizer	SGD
learning_rate	0.1
num_early_stop	20
num_decay_epoch	5
lr_decay	0.9
min_lr	0.00001
pos_dim	30
pred_ind_dim	30
emb_dropout	0.5
hid_dim	512
num_heads	8
d_k	64
d_v	64
d_ff	2048

Table 11: Basic hyperparameters applied in the experiments.

B Experiments

B.1 Dependency Parsers and POS Taggers

Table 10 shows the POS tagger and dependency parser evaluation results on each treebank in UPB v2. For `Japanese-GSDLUW`, `French-Rhapsodie`, and `English-EWT` treebanks, we train the POS taggers and dependency parsers from scratch using Stanza (Qi et al., 2020) with a 0.0005 learning rate, $70,000$ max steps, and $10,000$ max steps before stopping. We measure the performance of POS taggers with the F1 score. Meanwhile, we measure the performance of dependency parsers with the unlabeled attachment score (UAS) and the labeled attachment score (LAS).

B.2 Hyperparameter Search

We take the representative models from transformer-based models (i.e., SAN-RPRs), GCN-based models (i.e., ARGCNs and SGCNs), GAT-based models (i.e., GATs), and BiLSTM-based model (i.e., BiLSTMs) to experiment with the optimizers. We experiment with SGD (Kiefer and Wolfowitz, 1952), Adam (Kingma and Ba, 2015), and AdamW (Loshchilov and Hutter, 2019) as the optimizer. We also try different learning rates for each optimizer, i.e., 0.1, 0.01, and 0.001. The SGD optimizer with a 0.1 learning rate works the best in all the representative models. Therefore, we apply this setting to the rest of our experiments. Table 11 summarizes the fixed hyperparameters we use throughout our experiments. Algorithm 1 shows the logic for model training. Below, we explain each hyperparameter:

1. `num_epochs`: Number of epochs for model training.
2. `batch_size`: Batch size for model training.
3. `optimizer`: Type of optimizer for model training.
4. `learning_rate`: Initial learning rate for model training.
5. `num_early_stop`: Stop the training if there is no improvement after a certain number of consecutive epochs.
6. `num_decay_epoch`: The upper limit of epoch before we start decaying the learning rate.
7. `lr_decay`: The ratio to decay the learning rate.
8. `min_lr`: Lower limit of the learning rate. We stop the model training if the learning rate falls below this threshold.
9. `pos_dim`: Dimension of POSE, o (Equation 1). If the dimension is 0, we will not concatenate o in the input layer.
10. `pred_ind_dim`: Dimension of PIE, p (Equation 1). If the dimension is 0, we will not concatenate p in the input layer.
11. `emb_dropout`: Dropout applied in the input layer (Equation 1).
12. `hid_dim`: The dimension of the node representation that the encoder accepts.
13. `num_heads`: Number of heads applied in the multi-head self-attention mechanism present in transformer-based models, GAT-based models, and ARGCNs.
14. `d_k`: The dimension of keys applied in transformer-based models.
15. `d_v`: The dimension of values applied in transformer-based models.
16. `d_ff`: The output dimension of the first linear transformation in transformer-based model's position-wise FFN.

The following sections will explain the hyperparameter search and hyperparameter values that

Algorithm 1 Pseudocode of the model training.

```
Require: num_early_stop, num_decay_epoch, min_lr, lr_decay, num_epochs, learning_rate
  best_f1 ← 0
  no_improvement ← 0
  for curr_epoch ← 1, num_epochs do
      curr_f1 ← train(learning_rate)
      if curr_f1 > best_f1 then
          best_f1 ← curr_f1
          no_improvement ← 0
      else
          no_improvement ← no_improvement + 1
          if no_improvement ≥ num_early_stop then
              break
          end if
          if curr_epoch > num_decay_epoch then
              learning_rate ← lr_decay * learning_rate
              if learning_rate < min_lr then
                  break
              end if
          end if
      end if
  end for
```

work best in each model. Table 12, Table 13, Table 14, and Table 15 describe the hyperparameter search for BiLSTM-based models, transformer-based models, GCN-based models, and GAT-based models, respectively. Due to the number of hyperparameters, we divide the hyperparameter search into groups indicated by the leftmost column, i.e., the column with a "No" header. We will search for the best combination between the hyperparameters in the same group. For example, in Table 13, `num_enc_layers` and `enc_dropout` belong to group 1, which means we experiment with different dropouts, i.e., 0.1, 0.2, 0.3, 0.4, and 0.5, for each number of layers, i.e., 1, 2, 3, and 4. Below, we explain each hyperparameter involved in the hyperparameter search.

1. `num_enc_layers`: Number of layers stacked together.

2. `enc_dropout`: Dropout applied in the models, including dropout applied in the encoder (Section 3.3).

3. `lstm_num_layers`: Number of BiLSTM layers stacked together.

4. `lstm_dropout_net`: Dropout applied in BiLSTMs, including dropout applied in the encoder (Section 3.3).

5. `gnn_activation`: Activation function used in GCN-based and GAT-based models, applied in the encoder (Section 3.3).

6. `gnn_activation_at_final_layer`: Whether to apply the activation function in the last layer, especially if we stack more than one layer.

7. `lstm_activation`: Activation function used in BiLSTMs (Section 3.3).

8. `lstm_hidden_size`: Hidden size of BiLSTMs. Since the network is bidirectional, the dimension of the final hidden representation is $2 \times$ `lstm_hidden_size`.

9. `deprel_dim`: Dimension of DRE, d (Equation 1). If the dimension is 0, we will not concatenate d in the input layer.

10. `abs_position_dim`: Dimension of APE, a, and SAPE, s (Equation 1).

11. `use_dep_abs_position`: Boolean value that indicates whether to concatenate SAPE, s, in the input layer (Equation 1).

12. `use_word_abs_position`: Boolean value that indicates whether to concatenate APE, a, in the input layer (Equation 1).

13. `att_dim`: Dimension of a trainable vector a in ARGCNs (Equation 7 in Jiang et al. (2021)).

14. `base_size`: Base size, B, in RGCNs (Equation 3 in Schlichtkrull et al. (2018)).

15. `rel_pos_dim`: Dimension of RPR in GCN-based and GAT-based models. We do not use this parameter for transformer-based models

because the RPR in transformer-based models must have the same dimension as d_k and d_v (Vaswani et al., 2017).

16. `num_embed_graph_heads`: Number of heads where we modify M matrix according to distance matrix, D, in GATEs (Equation 3 in Ahmad et al. (2021b)). For the rest of the heads, we apply a zero matrix to M, connecting all the nodes in the graph.

17. `max_tree_dists`: The δ parameter that is applied to each head in GATEs (Equation 3 in Ahmad et al. (2021b)). The length of this parameter must equal `num_embed_graph_heads`.

18. `max_relative_positions`: The maximum absolute value for relative position (k in Shaw et al. (2018)) or structural relative position (r in Section Wang et al. (2019b)).

19. `use_dep_rel_pos`: The boolean value that indicates whether to incorporate SRPR in the edge representation.

20. `use_word_rel_pos`: The boolean value that indicates whether to incorporate RPR in the edge representation.

21. `deprel_edge_dim`: Dimension of representation for dependency relation, d_r, when we use B-DR.

22. `deparc_edge_dim`: Dimension of representation for dependency direction, d_d, when we use B-DR.

23. `deprel_ext_edge_dim`: Dimension of dependency relation representation when we use A-DR.

There are two ways of generating APE, SAPE, RPR, and SRPR, i.e., using learned positional embedding (Gehring et al., 2017) and using sine and cosine functions (Vaswani et al., 2017). We conduct preliminary experiments and find that sine and cosine functions work better than learned positional embedding. Therefore, we generate the representation for each type of position in the experiments using sine and cosine functions.

B.2.1 BiLSTM-Based Models

Table 12 shows the hyperparameter search for BiLSTM-based models.

No	Hyperparameter	Value
1	`lstm_num_layers` `lstm_dropout_net`	1, 2, 3, **4** 0.1, 0.2, **0.3**, 0.4, 0.5
2	`lstm_activation`	**ReLU**, Leaky ReLU, ELU
3	`lstm_hidden_size`	**256**, 512

4	`deprel_dim`
	`abs_position_dim`
	`use_dep_abs_position`
		0, **30**
		0, **30**
		T

Table 12: Hyperparameter search in BiLSTM-based models. The search is divided into groups shown in the "No" header. We search for the best combination of hyperparameters in the same group. The bold values indicate the results of the hyperparameter search.

B.2.2 Transformer-Based Models

Table 13 shows the hyperparameter search for transformer-based models.

B.2.3 GCN-Based Models

Table 14 shows the hyperparameter search for GCN-based models.

B.2.4 GAT-Based Models

Table 15 shows the hyperparameter search for GAT-based models.

B.3 Computational Resource

We use Tesla P100 to train the models. Training time for GCN-based and GAT-based models takes around 5 hours. Meanwhile, training time for BiLSTM-based and transformer-based models takes around 10 hours. Hyperparameter search in GCN-based models costs around 775 GPU hours. Hyperparameter search in GAT-based models costs around 910 GPU hours. Hyperparameter search in transformer-based models costs around 1,720 GPU hours. Hyperparameter search in BiLSTM-based models costs around 290 GPU hours. After searching for the best hyperparameter setting for each model, we run the training five times for each model, spending around 1,075 GPU hours. Therefore, in total, we spend approximately 4,770 hours.

C Supporting Results

C.1 Transformer-Based Models

C.1.1 Comparison

Table 16 shows the detailed comparison of transformer-based models in each language. We calculate each model's superiority score (SC) based on the model performance in target languages. We

No	Hyperparameter	Value
General		
1	num_enc_layers enc_dropout	1, 2, **3**, 4 0.1, **0.2**, 0.3, 0.4, 0.5
Transformers		
1	deprel_dim	0, **30**
GATEs		
1	num_embed_graph_heads max_tree_dists	4 <1, 1, 2, 2>, <2, 2, 4, 4>, <**4, 4, 8, 8**>, <1, 2, 4, 8>
2	deprel_dim abs_position_dim <use_dep_abs_position, use_word_abs_position>	0, **30** 0, **30** <**T, F**>, <F, T>
SAN-RPRs		
1	max_relative_positions	1, 2, 4, 8, **16**
2	deprel_dim	0, **30**
SAN-SAPRs		
1	deprel_dim	**0**, 30
SAN-SRPRs		
1	max_relative_positions	1, **2**, 4, 8, 16
2	deprel_dim	**0**, 30
Trans-SRPRs		
1	deprel_dim	0, **30**
Trans-SRPR-DRs		
1	deprel_dim abs_position_dim use_word_abs_position use_dep_rel_pos <deprel_edge_dim, deparc_edge_dim, deprel_ext_edge_dim>	0, **30** 0, **30** T **T**, F <32, 32, 0>, <48, 16, 0>, <56, 8, 0>, <**60, 4, 0**>, <62, 2, 0>, <63, 1, 0> <0, 0, 64>
SAPR-RPRs		
1	deprel_dim	**0**, 30
SAPR-RPR-DRs		
1	deprel_dim abs_position_dim use_dep_abs_position use_word_rel_pos <deprel_edge_dim, deparc_edge_dim, deprel_ext_edge_dim>	**0**, 30 0, **30** T T, **F** <32, 32, 0>, <48, 16, 0>, <56, 8, 0>, <**60, 4, 0**>, <62, 2, 0>, <63, 1, 0> <0, 0, 64>

Table 13: Hyperparameter search in transformer-based models. The search is divided into groups shown in the "No" header. We search for the best combination of hyperparameters in the same group. The bold values indicate the results of the hyperparameter search.

No	Hyperparameter	Value
SGCNs		
1	num_enc_layers enc_dropout	1, 2, **3**, 4 0.1, **0.2**, 0.3, 0.4, 0.5
2	gnn_activation gnn_activation_at_final_layer	**ReLU**, Leaky ReLU, ELU T, **F**
3	deprel_dim abs_position_dim <use_dep_abs_position, use_word_abs_position>	0, **30** **0**, 30 <T, F>, <F, T>
RGCNs		
1	num_enc_layers enc_dropout	**1**, 2, 3, 4 0.1, 0.2, 0.3, 0.4, **0.5**
2	gnn_activation gnn_activation_at_final_layer	**ReLU**, Leaky ReLU, ELU **T**, F
3	base_size	1, **2**, 4, 8, 16, 32, 80
4	deprel_dim abs_position_dim <use_dep_abs_position, use_word_abs_position>	**0**, 30 0, **30** <**T, F**>, <F, T>
ARGCNs		
1	num_enc_layers enc_dropout	**1**, 2, 3, 4 0.1, 0.2, 0.3, 0.4, **0.5**
2	gnn_activation gnn_activation_at_final_layer	**ReLU**, Leaky ReLU, ELU **T**, F
3	deprel_ext_edge_dim att_dim rel_pos_dim use_word_rel_pos	1, 2, 4, **8**, 16 1, 2, 4, 8, **16** **64**, 128 T
4	deprel_dim abs_position_dim use_dep_abs_position	0, **30** **0**, 30 T

Table 14: Hyperparameter search in GCN-based models. The search is divided into groups shown in the "No" header. We search for the best combination of hyperparameters in the same group. The bold values indicate the results of the hyperparameter search.

allocate 2 points if the model achieves the highest F1 score or 1 point if the model achieves the second-highest F1 score for a specific language.

C.1.2 Ablation Study of SAN-RPRs

We experiment with removing either DRE, d, POSE, o, or PIE, p, from the node representation in SAN-RPRs. Table 17 shows the results of the ablation study. Removing either DRE or PIE from the node representation reduces the performance of SAN-RPRs. However, SAN-RPRs without POSE perform better in most languages.

C.2 Best Models

C.2.1 Comparison with SAN-RPRs w/o POSE

In Table 5, according to the superiority score, we can see that SAPR-RPRs perform best in more languages than SAN-RPRs, even though the average F1 score of SAN-RPRs is better than SAPR-RPRs. However, as discussed before, SAN-RPRs without POSE perform better in most languages than SAN-RPRs with POSE, which we use for comparison in Table 5. Therefore, in Table 18, we re-compare SAN-RPRs without POSE with the other best models. After removing POSE from SAN-RPRs, we find that the model performs best in more languages than SAPR-RPRs with 18 and 16 superiority scores, respectively.

C.2.2 Fine-Tuned Models

We fine-tune the multilingual BERT (mBERT) in SAN-RPRs and SAPR-RPRs. Table 19 compares the average F1 scores of models with frozen and fine-tuned mBERT. Overall, fine-tuning increases the performance of both models. However, in the fine-tuned models, the variability of the average F1 score in each run increases, indicated by the higher standard deviation in fine-tuned mBERT. This is expected as when we fine-tune the mBERT, many parameters from mBERT are involved in the training process, increasing the randomness variable in model training. The behavior of the models is similar before and after the fine-tuning. The fine-tuned SAN-RPRs perform better than SAPR-RPRs with 54.01% and 53.82% average F1 scores, respectively.

No	Hyperparameter	Value
General		
1	gnn_activation gnn_activation_at_final_layer	ReLU, **Leaky ReLU**, ELU T, **F**
GATs		
1	num_enc_layers enc_dropout	1, **2**, 3, 4 **0.1**, 0.2, 0.3, 0.4, 0.5
2	deprel_dim abs_position_dim <use_dep_abs_position, use_word_abs_position>	0, **30** 0, **30** <**T, F**>, <F, T>
SHGNs		
1	num_enc_layers enc_dropout	1, **2**, 3, 4 0.1, 0.2, **0.3**, 0.4, 0.5
2	deprel_ext_edge_dim rel_pos_dim use_word_rel_pos	**16**, 32, 64, 128 **16**, 32, 64, 128 T
3	deprel_dim abs_position_dim use_word_rel_pos use_dep_abs_position	**0**, 30 **0**, 30 T, **F** T
4	<deprel_edge_dim, deparc_edge_dim, deprel_ext_edge_dim>	<8, 8, 0>, <12, 4, 0>, <14, 2, 0>, <15, 1, 0>, <**0, 0, 16**>
TAGATs		
1	num_enc_layers enc_dropout	1, **2**, 3, 4 0.1, 0.2, **0.3**, 0.4, 0.5
2	deprel_ext_edge_dim rel_pos_dim use_word_rel_pos	**16**, 32, 64, 128 16, 32, 64, **128** T
3	deprel_dim abs_position_dim use_word_rel_pos use_dep_abs_position	**0**, 30 0, **30** **T**, F T
4	<deprel_edge_dim, deparc_edge_dim, deprel_ext_edge_dim>	<8, 8, 0>, <12, 4, 0>, <14, 2, 0>, <15, 1, 0>, <**0, 0, 16**>
KBGATs		
1	num_enc_layers enc_dropout	1, **2**, 3, 4 0.1, **0.2**, 0.3, 0.4, 0.5
2	deprel_ext_edge_dim rel_pos_dim use_word_rel_pos	16, **32**, 64, 128 16, **32**, 64, 128 T
3	deprel_dim abs_position_dim use_word_rel_pos use_dep_abs_position	0, **30** **0**, 30 **T**, F T
4	<deprel_edge_dim, deparc_edge_dim, deprel_ext_edge_dim>	<16, 16, 0>, <**24, 8, 0**>, <28, 4, 0>, <30, 2, 0>, <31, 1, 0>, <0, 0, 32>

Table 15: Hyperparameter search in GAT-based models. The search is divided into groups shown in the "No" header. We search for the best combination of hyperparameters in the same group. The bold values indicate the results of the hyperparameter search.

	GATEs	Transformers	SAN-RPRs	SAN-SAPRs	SAN-SRPRs	Trans-SRPRs	SAPR-RPRs	Trans-SRPR-DRs	SAPR-RPR-DRs
EN	78.96±0.31	76.16±0.51	78.26±0.40	75.93±0.47	78.27±0.50	79.03±0.32	78.11±0.42	79.85±0.21	79.83±0.19
AVG	52.57±0.23	52.07±0.21	**52.73±0.40**	51.98±0.10	51.51±0.27	52.21±0.30	52.64±0.38	50.60±0.14	50.69±0.21
TA	36.72±0.89	35.32±1.56	37.96±1.75	38.07±1.47	35.93±2.16	35.74±1.47	**39.57±1.18**	32.23±1.14	32.91±1.22
HI	**48.56±0.37**	47.40±0.18	47.51±0.62	45.07±0.52	45.70±0.63	47.80±0.57	45.04±0.35	48.04±0.45	47.57±0.65
ZH	48.09±0.64	48.86±0.31	50.37±1.17	50.06±0.15	46.96±0.54	46.51±0.53	**50.96±0.88**	43.63±1.09	43.12±0.87
JA	37.45±0.86	**38.38±0.37**	37.69±0.99	37.91±0.88	36.29±1.60	36.46±1.20	34.78±1.29	33.86±0.61	33.56±1.47
VI	28.01±0.91	28.25±0.57	28.69±0.79	**29.70±0.50**	27.70±0.61	27.34±0.59	29.10±0.45	25.72±0.22	25.37±0.23
KO	43.44±0.90	43.13±2.26	42.61±1.84	**46.09±1.02**	43.38±0.84	43.62±0.57	45.24±1.23	39.34±1.19	40.76±0.51
ID	57.88±0.47	55.36±0.42	58.78±1.09	57.95±0.56	59.79±0.27	58.41±0.35	**59.97±0.53**	53.41±0.69	52.89±0.25
HU	**50.69±0.25**	50.13±0.85	49.76±0.35	49.68±0.56	49.28±0.20	50.60±0.49	49.08±0.34	49.51±0.47	49.31±0.72
RO	53.86±0.57	52.90±0.39	54.23±0.67	52.11±0.23	52.44±0.33	53.75±0.55	**54.46±0.52**	52.50±0.31	52.50±0.49
FR	61.67±0.30	60.23±0.39	**62.19±0.41**	60.52±0.27	60.34±0.37	61.30±0.50	62.11±0.47	60.82±0.20	60.91±0.28
MR	37.49±1.74	**42.85±2.67**	41.06±2.89	41.22±1.65	37.37±1.27	36.58±1.68	40.36±2.20	36.75±1.67	37.40±1.56
UK	58.89±0.53	58.83±0.20	58.92±0.26	58.87±0.48	58.23±0.69	58.93±0.72	**59.36±0.72**	57.94±0.42	58.23±0.39
PT	65.49±0.24	64.05±0.26	66.05±0.21	64.76±0.48	65.16±0.41	65.34±0.15	**66.49±0.33**	64.61±0.24	64.84±0.17
IT	57.52±0.38	57.42±0.43	**58.11±0.33**	56.35±0.45	56.09±0.33	57.42±0.32	57.80±0.39	56.60±0.31	56.72±0.42
ES	63.36±0.24	62.04±0.42	**63.71±0.33**	61.53±0.31	62.70±0.20	63.47±0.27	63.62±0.25	61.78±0.14	61.68±0.22
CS	**57.69±0.39**	56.19±0.48	56.87±0.27	55.27±0.57	56.58±0.27	57.61±0.34	55.80±0.51	56.44±0.26	56.35±0.49
EL	60.37±0.59	59.03±0.34	**60.59±0.23**	57.30±0.69	58.98±0.82	59.31±0.30	60.23±0.40	57.89±0.33	58.33±0.33
FI	55.00±0.34	54.72±0.33	**55.58±0.42**	54.74±0.47	53.91±0.37	54.86±0.31	55.29±0.54	54.11±0.17	54.15±0.25
RU	59.36±0.24	59.31±0.47	60.18±0.44	60.33±0.37	59.09±0.30	59.29±0.40	**61.13±0.50**	58.60±0.39	58.56±0.25
NL	**63.73±0.53**	62.48±0.76	62.84±0.37	61.75±0.26	61.73±0.41	63.07±0.49	62.22±0.58	62.23±0.20	62.43±0.39
TE	**46.32±0.98**	44.94±1.38	44.66±2.00	41.64±1.81	41.88±1.17	46.10±1.10	43.88±1.57	42.48±1.28	43.09±1.26
DE	**58.88±0.40**	58.29±0.70	56.86±0.32	58.39±0.34	58.14±0.30	58.84±0.30	56.98±1.13	58.21±0.26	58.30±0.34
PL	**58.73±0.64**	57.58±0.25	57.67±0.36	56.25±0.35	57.14±0.40	58.60±0.29	57.28±0.46	57.18±0.62	56.85±0.64
SC	15	4	13	8	1	7	20	1	0
PR	12.2M	12.2M	12.2M	12.2M	12.1M	12.2M	12.2M	12.2M	12.2M

Table 16: F1 scores (%) of transformer-based models evaluated on UPB v2 test set with predicted parsers. The bold score and underlined score indicate the highest and second-highest scores. AVG indicates the average F1 scores of a specific model evaluated in target languages. PR and SC are the number of parameters and the superiority score of each model.

	base	w/o DRE	w/o POSE	w/o PIE
EN	78.26±0.40	77.56±0.41	78.32±0.32	77.66±0.38
AVG	**52.73±0.40**	51.65±0.28	**52.73±0.32**	51.54±0.36
TA	**37.96±1.75**	37.59±1.95	36.53±1.32	32.03±1.68
HI	47.51±0.62	42.22±0.85	**47.69±1.10**	43.89±0.81
ZH	50.37±1.17	50.42±0.38	**50.81±0.51**	49.83±0.54
JA	**37.69±0.99**	33.25±2.01	36.07±1.20	30.67±2.90
VI	28.69±0.79	29.18±1.05	28.30±0.92	**29.34±0.96**
KO	42.61±1.84	**42.73±1.52**	42.06±1.50	40.13±0.65
ID	**58.78±1.09**	58.41±0.89	57.77±0.83	57.06±0.78
HU	49.76±0.35	48.25±0.62	**50.02±0.42**	49.27±0.86
RO	54.23±0.67	54.11±0.37	**54.94±0.50**	54.19±0.76
FR	62.19±0.41	61.74±0.32	**62.36±0.59**	61.85±0.60
MR	41.06±2.89	39.45±1.76	**41.83±3.50**	41.15±2.62
UK	58.92±0.26	58.75±0.94	58.84±0.59	**59.64±0.15**
PT	66.05±0.21	**66.67±0.53**	66.65±0.44	65.46±0.44
IT	58.11±0.33	57.43±0.24	**58.65±0.49**	58.34±0.32
ES	63.71±0.33	63.18±0.33	**64.07±0.51**	63.63±0.34
CS	56.87±0.27	54.94±0.40	**57.20±0.14**	56.40±0.28
EL	60.59±0.23	59.40±0.23	**60.86±0.56**	59.87±1.11
FI	**55.58±0.42**	54.78±0.25	55.40±0.16	55.18±0.24
RU	**60.18±0.44**	60.05±0.67	60.12±0.36	60.00±0.28
NL	62.84±0.37	60.97±0.84	**63.21±0.38**	60.76±0.68
TE	**44.66±2.00**	42.41±2.63	43.46±2.02	44.40±3.32
DE	56.86±0.32	55.34±1.44	**57.70±0.43**	54.36±0.52
PL	57.67±0.36	56.68±0.63	**58.16±0.35**	58.01±0.31

Table 17: F1 scores (%) of SAN-RPRs with certain embedding removed from the node representation evaluated on UPB v2 test set with predicted parsers. The bold score and underlined score indicate the highest and second-highest scores. AVG indicates the average F1 scores of a specific model evaluated in target languages.

	SAN-RPRs (w/o POSE)	SAPR-RPRs	SGCNs	GATs	TAGATs	BiLSTMs
EN	78.32±0.32	78.11±0.42	79.94±0.27	79.81±0.19	79.07±0.19	76.86±0.34
AVG	52.73±0.32	52.64±0.38	52.52±0.38	52.66±0.14	**52.78±0.14**	51.85±0.09
TA	36.53±1.32	**39.57±1.18**	34.32±1.12	35.08±0.58	35.68±1.28	34.19±1.22
HI	47.69±1.10	45.04±0.35	48.24±0.61	**48.25±0.33**	47.65±0.33	46.63±0.38
ZH	50.81±0.51	**50.96±0.88**	45.77±0.67	46.12±0.39	46.80±0.64	47.56±0.89
JA	36.07±1.20	34.78±1.29	37.43±0.28	37.99±0.52	**39.30±0.73**	37.40±0.61
VI	28.30±0.92	**29.10±0.45**	27.95±0.56	28.06±0.59	28.31±0.55	28.18±0.88
KO	42.06±1.50	**45.24±1.23**	42.92±0.64	43.22±0.56	44.57±0.24	41.77±1.61
ID	57.77±0.83	**59.97±0.53**	58.54±0.82	58.33±0.69	59.11±0.87	56.11±0.84
HU	50.02±0.42	49.08±0.34	50.76±0.41	**51.10±0.51**	50.90±0.37	50.64±0.39
RO	**54.94±0.50**	54.46±0.52	53.57±0.47	54.12±0.49	53.60±0.45	53.26±0.34
FR	**62.36±0.59**	62.11±0.47	60.93±0.38	61.64±0.44	61.13±0.22	61.22±0.27
MR	**41.83±3.50**	40.36±2.20	40.97±3.40	38.06±0.13	39.26±1.20	37.18±2.28
UK	58.84±0.59	59.36±0.72	59.66±0.76	59.49±0.56	**59.72±0.31**	58.96±0.07
PT	**66.65±0.44**	66.49±0.33	65.62±0.43	65.99±0.32	65.61±0.15	64.40±0.33
IT	**58.65±0.49**	57.80±0.39	57.43±0.42	58.00±0.42	57.34±0.34	58.02±0.27
ES	64.07±0.51	63.62±0.25	63.87±0.61	**64.29±0.36**	63.91±0.27	62.48±0.29
CS	57.20±0.14	55.80±0.51	57.95±0.52	**58.02±0.21**	57.62±0.28	56.59±0.36
EL	**60.86±0.56**	60.23±0.40	60.56±0.69	60.74±0.48	**60.86±0.34**	59.76±0.45
FI	**55.40±0.16**	55.29±0.54	54.87±0.40	54.62±0.32	54.88±0.20	54.62±0.20
RU	60.12±0.36	**61.13±0.50**	59.98±0.34	60.14±0.16	60.30±0.22	59.73±0.25
NL	63.21±0.38	62.22±0.58	62.94±0.21	**63.53±0.64**	62.97±0.37	62.47±0.36
TE	43.46±2.02	43.88±1.57	46.08±1.07	**46.96±1.49**	**46.96±1.82**	45.95±0.51
DE	57.70±0.43	56.98±1.13	**58.61±0.30**	58.52±0.26	58.52±0.18	57.72±0.23
PL	58.16±0.35	57.28±0.46	**59.08±0.31**	59.00±0.44	58.92±0.52	57.73±0.23
SC	18	16	-	-	-	-

Table 18: F1 scores (%) of best models evaluated on UPB v2 test set with predicted parsers. The bold score and underlined score indicate the highest and second-highest scores. AVG indicates the average F1 scores of a specific model evaluated in target languages. SC indicates the superiority score of each model.

	Frozen mBERT		Fine-Tuned mBERT	
	SAN-RPRs	SAPR-RPRs	SAN-RPRs	SAPR-RPRs
EN	78.26±0.40	78.11±0.42	79.37±0.60	79.04±0.57
AVG	52.73±0.40	52.64±0.38	54.01±0.95	53.82±1.17

Table 19: F1 scores (%) of SAN-RPRs and SAPR-RPRs with frozen mBERT and fine-tuned mBERT. AVG indicates the average F1 scores of a specific model evaluated in target languages.