Typology-Aware Multilingual Morphosyntactic Parsing with Joint Abstract Node Modeling

Anonymous ACL submission

Abstract

The UniDive 2025 Morphosyntactic Parsing (MSP) shared task introduces a representation unifying dependency structure, morphological features, and unrealized arguments. Unlike Universal Dependencies, MSP encodes abstract nodes (e.g., dropped subjects, implicit pronouns) as labels projected onto content words, which standard UD parsers cannot model. To address this challenge, in this paper we present a multilingual, typology-aware joint system integrating word-type prediction, content-only parsing, morphological tagging, and an abstract-node component within a single architecture. The model combines the baseline joint framework with typology-conditioned adapters and progressive weighting for abstract supervision. On the MSP test set, our model outperforms the leading submission by 3.23 percentage points in MSLAS, 3.35 in LAS, and 1.78 in FEATS macro F1, demonstrating the effectiveness of typology-sensitive multi-task learning in MSP.

1 Introduction

Syntactic parsing is a task in natural language processing, and the Universal Dependencies (UD) (de Marneffe et al., 2021) project provides a framework for representing syntax as dependency trees. Recently, the UniDive Morphosyntactic Parsing (MSP) Shared Task (Goldman et al., 2025), inspired by work (Bārzdiņš et al., 2007; Nivre et al., 2022; Goldman and Tsarfaty, 2022) that moves beyond rigid word-based representations of syntax and morphology, introduced a representation that models morphosyntax in a unified and harmonized manner. Unlike UD, MSP distinguishes between content and function words and reduces the parsing task to identifying dependencies among content nodes, reflecting MSP's typologically diverse setting and motivating typology-aware adaptation of morphosyntactic variation. Participating systems are required to jointly predict (i) a dependency structure over content words, (ii) morphological feature bundles, and (iii) abstract nodes representing unrealized arguments, such as dropped subjects, implicit pronouns, and argument gaps.

UD parsers, including UDify (Kondratyuk and Straka, 2019) and UDapter (Üstün et al., 2020, 2022), assume that syntactic structure is realized over surface tokens. MSP breaks the token-centric assumption: function words are not tree nodes, and unrealized arguments contribute dependency relations and morphological features only through projections onto content words (Goldman et al., 2025). More broadly, reflects a challenge for multilingual parsing in languages with ellipsis, pro-drop, and morphology-driven argument omission, where syntactic structure and surface realization do not align. As a result, standard UD parsers are not suitable for MSP, and only a number of prior approaches incorporate typology-aware mechanisms important for multilingual generalization.

We propose a new MSP architecture[1] that builds on the joint modeling framework of Amésticа et al. (2025). We extend this framework by incorporating typology-aware multilingual conditioning (Acar and Eryiğit, 2025) and by explicitly modeling abstract nodes within the parsing architecture, yielding a fully joint and typology-sensitive MSP system. Experimental results show that the proposed system outperforms the previous state of the art in both macro-averaged performance (MSLAS $+3.23$, LAS $+3.35$, FEATS $+1.78$) and language-specific evaluation, achieving higher MSLAS and LAS scores across all MSP languages.

Sections 2–4 introduce the MSP formalism, the proposed architecture, and abstract-node supervision. Sections 5–7 present the experimental setup, results, and analysis, while Section 8 concludes the paper.

[1] Software available from https://github.com/closedforblindreview.

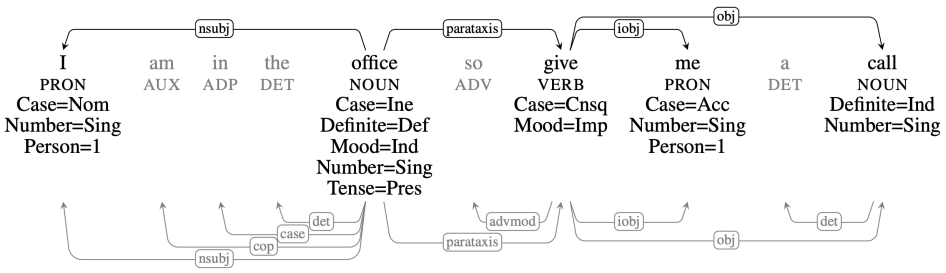

Figure 1: An example of a morphosyntactic tree above the sentence, with the full UD tree below for reference. Function words do not participate in the morphosyntactic tree and are coloured grey. (Goldman et al., 2025)

2 Background

As illustrated in Figure 1, the morphosyntactic tree in MSP is defined over overt content words, while functional material is layered onto these tokens through morphological feature bundles. Function words do not participate in the tree structure itself. Although abstract nodes are not explicitly visualized in the figure, MSP further extends this representation by encoding unrealized arguments, such as dropped subjects and implicit pronouns, as abstract projections associated with content words, contributing additional dependency relations and morphological features that are not recoverable from surface syntax. This design departs fundamentally from UD, where all syntactic and functional elements are represented as explicit tree nodes (de Marneffe et al., 2021). The UniDive (Savary et al., 2024) MSP Shared Task is the first multilingual benchmark to formalize this representation and to evaluate systems on the joint coherence of syntax and morphology (Goldman et al., 2025).

Traditional dependency parsers (both transition-based (Nivre, 2003; Nivre et al., 2006; Hall et al., 2007) and graph-based (McDonald et al., 2005)) assume a one-to-one correspondence between surface tokens and syntactic nodes, with morphology predicted directly at the token level. Modern neural UD parsers, including biaffine models (Dozat and Manning, 2016) and multilingual systems such as UDify (Kondratyuk and Straka, 2019) and UDapter (Üstün et al., 2020, 2022), largely retain these assumptions. This formulation does not align with MSP, where unrealized arguments are represented as abstract nodes that form dependency relations and carry morphological features, but are realized exclusively through projections onto overt content words. As a result, they cannot be directly incorporated into standard token-based parsing and tagging schemes. Consequently, standard UD parsers cannot natively produce valid MSP outputs without substantial task-specific modifications, as also noted in the findings (Goldman et al., 2025).

The UniDive 2025 MSP Shared Task attracted a small number of task-specific systems, reflecting both the novelty and difficulty of the formulation (Goldman et al., 2025). The strongest submission employed a joint multitask architecture integrating word-type prediction, content-only dependency parsing, and morphological tagging within a shared encoder (Amestica et al., 2025). Another submission incorporated typology-aware representations and content/function distinctions, but trained word-type classification separately from parsing and tagging (Acar and Eryiğit, 2025), highlighting the importance of joint modeling in MSP. The shared task also included a few-shot prompting baseline based on large language models, which underperformed dedicated MSP architectures. Overall, prior systems either lack typology-aware conditioning within the core encoder or do not explicitly model abstract nodes as structured prediction targets.

MSP is inherently multilingual and typologically diverse. Prior work in typology-aware NLP, including URIEL features (Littell et al., 2017) and hypernetwork-based adapters for UD parsing (Üstün et al., 2020, 2022), has shown that conditioning on language typology can improve cross-lingual generalization. However, no previous work integrates typology-aware conditioning into a fully joint MSP model that simultaneously predicts dependency structure, morphology, and abstract nodes within a single architecture. Abstract nodes are central to MSP, encoding unrealized arguments together with their syntactic relations and morphological features. Addressing this gap benefits from joint modeling, typology-sensitive adaptation, and explicit abstract-node prediction, which is the focus of the present work.

3 System Architecture

This section describes the proposed multilingual, multi-task architecture for MSP, detailing how input representations, typological information, and task-specific components are integrated into a joint system. An overview of the full system architecture is provided in Appendix A.

3.1 Model Input & Typology Incorporation

The model operates on the base rows of the MSP CoNLL-U format (Appendix B of Goldman et al. (2025)). During training and inference, sentences are represented only as sequences of overt tokens. Functional and abstract nodes are never passed as separate encoder time steps. Instead, functional nodes are collapsed into word-type supervision and abstract nodes are projected onto their associated base token as supervisory labels for abstract presence, relation and morphological layering.

Input Representation Each sentence is encoded with a stacked embedding module combining a multilingual Transformer encoder and optional character-level information. Concretely, we use `XLM-RoBERTa` as a contextual word embedding layer and optionally augment it with a character-based BiLSTM encoder. The resulting token embeddings are concatenated and passed through a layer-normalization step and a small shared feed-forward block that refines the representations before they are consumed by the task heads:

$$\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^D, \ \mathbf{z}_i = \mathrm{FFN}(\mathbf{x}_i) \in \mathbb{R}^D. \quad (1)$$

Alongside token embeddings, each sentence carries a language identifier and a vector of URIEL typological features (Littell et al., 2017). The language identifier is used to index a learned language embedding, while the URIEL vector provides a compact, hand-crafted description of syntactic and phonological properties of the language. Importantly, only base tokens appear as positions in the sequence; abstract-node rows and functional-node stand-ins are used only to derive supervision labels and are never treated as separate tokens in the encoder.

During training, each token also carries a gold `word_type` label (content or function token). These labels are used directly to supervise the word-type classifier and implicitly to derive content-only masks for parsing and morphological tagging. At inference time, the same gating is driven by the model's predicted word-type labels.

Typology Encoder Typological information is injected into the architecture via a dedicated *TypologyEncoder*. For each sentence, the encoder takes as input (i) a language ID embedding and (ii) an optional URIEL feature vector. The URIEL vector is first projected through a linear layer and nonlinearity, and then concatenated with the language embedding. A small MLP maps this concatenated representation to a fixed-dimensional language vector h_{lang}:

$$\mathbf{e}_l = \mathbf{E}_{\text{lang}}[l], \quad (2)$$

$$\tilde{\mathbf{u}}_l = \tanh(\mathbf{W}_u \mathbf{u}_l + \mathbf{b}_u), \quad (3)$$

$$\mathbf{h}_{\text{lang}} = \mathrm{MLP}\big([\mathbf{e}_l; \tilde{\mathbf{u}}_l]\big). \quad (4)$$

This typology representation is shared across tasks. It is used to condition language-specific adapters, and it can optionally be trained with an auxiliary URIEL reconstruction loss. In that auxiliary task, a linear layer predicts the URIEL features from h_{lang}, and a masked binary cross-entropy loss is applied only to observed features, encouraging the model to learn a compact, task-informed language representation without being forced to match missing or uncertain typological entries:

$$\mathcal{L}_{\text{typ}} = \frac{1}{\sum_k m_k} \sum_k m_k \cdot \mathrm{BCE}(\hat{u}_{l,k}, u_{l,k}). \quad (5)$$

When this auxiliary objective is disabled, we set $\lambda_{\text{typ}} = 0$ in the total loss.

Contextual Hypernetwork Adapters To make the encoder sensitive to language-specific structure while sharing most parameters across languages, we adopt UDapter-style contextual adapters. Two separate adapters are instantiated: a *parser adapter* and a *tagger adapter*. Both maintain shared base projection matrices and generate small, language-conditioned *deltas* using hypernetworks conditioned on h_{lang}. Conceptually:

$$\tilde{\mathbf{z}}_i^{\text{par}} = A_{\text{par}}(\mathbf{z}_i, \mathbf{h}_{\text{lang}}), \quad (6)$$

$$\tilde{\mathbf{z}}_i^{\text{tag}} = A_{\text{tag}}(\mathbf{z}_i, \mathbf{h}_{\text{lang}}). \quad (7)$$

The resulting adapter is applied in a residual fashion on top of the shared encoder representation. The parser adapter is used exclusively before the dependency parsing head, while the tagger adapter is shared by the word-type, morphological and abstract-node heads. A scaling factor keeps the

deltas small at initialization so that the adapters initially behave close to identity and gradually learn language-specific deviations.

3.2 Joint Architecture Components

The architecture is organized as a joint model with several task heads built on top of shared, typology-adapted representations. The main components correspond to MSP's core dimensions: word type, content-only dependency structure, content-only morphological features, and abstract-node structure layered on top of base tokens.

Word-Type Classification Word-type classification is implemented as a sequence labeling task. The typology-adapted tagger representation is passed through a BiLSTM layer followed by locked dropout and a linear classifier that predicts a binary label (*content* vs. *function*) for each token:

$$\mathbf{h}_i^{\mathrm{wt}} = \mathrm{BiLSTM}(\tilde{\mathbf{z}}_i^{\mathrm{tag}}), \tag{8}$$

$$\mathbf{p}_i^{\mathrm{wt}} = \mathrm{softmax}(\mathbf{W}_{\mathrm{wt}}\mathbf{h}_i^{\mathrm{wt}} + \mathbf{b}_{\mathrm{wt}}). \tag{9}$$

Class weights are estimated on the fly from label frequencies in order to mitigate imbalances. These predictions later serve as the primary gating signal for dependency parsing and morphological feature prediction.

Dependency Parsing The dependency parser is a graph-based CRF parser trained on content tokens only. After the parser adapter is applied, separate MLPs generate dependency and head representations:

$$\mathbf{d}_i = f_{\mathrm{dep}}(\tilde{\mathbf{z}}_i^{\mathrm{par}}), \tag{10}$$

$$\mathbf{h}_j = f_{\mathrm{head}}(\tilde{\mathbf{z}}_j^{\mathrm{par}}). \tag{11}$$

Arc scores are computed bilinearly:

$$s_{ij}^{\mathrm{arc}} = \bar{\mathbf{d}}_i^{\top}\mathbf{U}_{\mathrm{arc}}\bar{\mathbf{h}}_j. \tag{12}$$

The arc scores parameterize a `DependencyCRF` distribution over trees. Training maximizes the log likelihood of the gold content-only tree:

$$\mathcal{L}_{\mathrm{arc}} = -\log p_{\mathrm{CRF}}(A^{\star} \mid s^{\mathrm{arc}}). \tag{13}$$

Relation prediction uses an analogous bilinear classifier:

$$\mathcal{L}_{\mathrm{rel}} = \frac{1}{|A^{\star}|}\sum_{(i\to j)\in A^{\star}} \mathrm{CE}\left(\mathrm{softmax}(s_{ij,\cdot}^{\mathrm{rel}}), y_{ij}^{\mathrm{rel}}\right). \tag{14}$$

The total parsing loss is:

$$\mathcal{L}_{\mathrm{par}} = \mathcal{L}_{\mathrm{arc}} + \mathcal{L}_{\mathrm{rel}}. \tag{15}$$

Morphological Tagging Morphological tagging is a multi-label classification task over feature–value items. A linear layer projects the tagger-adapted representation:

$$\mathbf{o}_i^{\mathrm{morph}} = \mathbf{W}_{\mathrm{morph}}\tilde{\mathbf{z}}_i^{\mathrm{tag}} + \mathbf{b}_{\mathrm{morph}}. \tag{16}$$

Independent sigmoid activations produce feature probabilities:

$$p_{i,k}^{\mathrm{morph}} = \sigma(o_{i,k}^{\mathrm{morph}}). \tag{17}$$

The loss is a binary cross-entropy term summed over gold feature–value items of content tokens:

$$\mathcal{L}_{\mathrm{morph}} = \frac{1}{N_{\mathrm{morph}}}\sum_{i\in\mathrm{content}}\sum_{k=1}^{K}\mathrm{BCE}(p_{i,k}^{\mathrm{morph}}, y_{i,k}). \tag{18}$$

Abstract Node Modeling The model includes a dedicated abstract-node prediction module responsible for modeling unrealized arguments as abstract nodes layered on top of base tokens. Unlike functional nodes, which determine gating, abstract nodes encode additional structure layered on top of base tokens. The model predicts four attributes for each token:

$$p_i^{\mathrm{pres}} = \sigma(\mathbf{w}_{\mathrm{pres}}^{\top}\tilde{\mathbf{z}}_i^{\mathrm{tag}} + b_{\mathrm{pres}}), \tag{19}$$

$$\mathbf{p}_i^{\mathrm{pos}} = \mathrm{softmax}(\mathbf{W}_{\mathrm{pos}}\tilde{\mathbf{z}}_i^{\mathrm{tag}} + \mathbf{b}_{\mathrm{pos}}), \tag{20}$$

$$\mathbf{p}_i^{\mathrm{dep}} = \mathrm{softmax}(\mathbf{W}_{\mathrm{dep}}\tilde{\mathbf{z}}_i^{\mathrm{tag}} + \mathbf{b}_{\mathrm{dep}}), \tag{21}$$

$$p_{i,k}^{\mathrm{feat}} = \sigma\big((\mathbf{W}_{\mathrm{feat}}\tilde{\mathbf{z}}_i^{\mathrm{tag}} + \mathbf{b}_{\mathrm{feat}})_k\big). \tag{22}$$

The full abstract-node objective is a combination of presence, position, dependency-label, and morphological-feature losses:

$$\mathcal{L}_{\mathrm{abs}} = \mathcal{L}_{\mathrm{pres}} + \mathcal{L}_{\mathrm{pos}} + \mathcal{L}_{\mathrm{dep}} + \mathcal{L}_{\mathrm{feat}}. \tag{23}$$

A detailed description of how abstract-node labels are extracted, how multiple abstract nodes are projected onto hosts, and how abstract structure is reconstructed at inference is provided in Section 4.

3.3 Joint Objective and Loss Weighting

The total training objective is a weighted combination of all task-specific losses:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{par}}\mathcal{L}_{\text{par}} + \lambda_{\text{morph}}\mathcal{L}_{\text{morph}} + \lambda_{\text{wt}}\mathcal{L}_{\text{wt}} + \lambda_{\text{abs}}(t)\,\mathcal{L}_{\text{abs}} + \lambda_{\text{typ}}\mathcal{L}_{\text{typ}}. \quad (24)$$

The abstract-node weight $\lambda_{\text{abs}}(t)$ follows the model's built-in ramp schedule:

$$\lambda_{\text{abs}}(t) = \lambda_{\text{abs}}^{\min} + \left(\lambda_{\text{abs}}^{\max} - \lambda_{\text{abs}}^{\min}\right) r(t), \quad (25)$$

$$r(t) = \left(\min(1, t/T)\right)^{\gamma}, \quad (26)$$

While the joint objective defines how abstract-node predictions interact with parsing and morphology during training, abstract nodes in MSP are not independent tokens but annotation-level projections tied to base words. As a result, their supervision, labeling, and reconstruction into valid MSP outputs motivate a task-specific treatment beyond standard parsing objectives. We describe this process in the next section.

4 Abstract Node Supervision and Inference

In the MSP setting, abstract nodes encode unrealized arguments as additional syntactic–morphological layers associated with a base token, rather than as independent tree nodes. They carry their own DEPREL, HEAD, and FEATS values, forming a parallel syntactic–morphological tier that cannot be captured by traditional UD parsers. The joint model therefore predicts abstract-node properties directly and reconstructs them into the final CoNLL-U output.

Extraction of Abstract Supervision Abstract supervision is extracted directly from the MSP-enhanced CoNLL-U data. We identify candidate abstract rows by their non-integer IDs of the form $X.Y$. Among these, only rows with non-empty FEATS, HEAD and DEPREL fields are treated as genuine abstract nodes representing unrealized arguments (e.g., dropped subjects, implicit pronouns, and argument gaps) with clear syntactic and morphological content. Rows with empty structural fields are discarded. Each valid abstract row is mapped back to a *host* base token X, which appears as a standard integer ID in the sentence.

Training Labels From the extracted abstract rows, we construct four label sets for each token:

- **abs_pres**: a binary indicator that is set to 1 if the token hosts at least one valid abstract node and to 0 otherwise;
- **abs_pos**: a categorical label over {ABOVE, BELOW} indicating whether the abstract node(s) appear above the host row, or below;
- **abs_deprel**: a single-label abstract dependency relation taken from the abstract row that best represents the unrealized argument or clausal structure attached to the host;
- **abs_feats**: a multi-label set of morphological feature–value items taken from the FEATS column of the abstract row, encoding properties such as person, number, tense or case associated with the unrealized argument.

These labels are stored in the training instances as token-level annotations and used to supervise the corresponding abstract-node heads described in the previous section.

Decoding and Reconstruction At inference time, the model first predicts word type, content-only dependency structure, and content-only morphological features, and then applies the abstract-node heads to obtain label predictions for each token. The predicted abstract structure is reconstructed into an MSP-style CoNLL-U representation in a post-processing step.

For each token whose predicted abstract presence exceeds a threshold, an abstract row is created and associated with that token. The predicted `abs_pos` label determines whether the abstract row is inserted immediately above or below the host row. Dependency relations and morphological features are filled from the corresponding abstract predictions, yielding a concrete CoNLL-U line.

Head indices for abstract nodes are assigned using a lightweight, annotation-aligned heuristic. For eight of the nine MSP languages, each abstract node is attached to the nearest content word occurring above the host token, reflecting the fact that unrealized arguments typically depend on an overt predicate or head in preceding context. Swedish constitutes the sole exception due to systematic annotation differences; for this language, we attach abstract nodes to the nearest verbal head in either direction, falling back to the host token if no suitable verb is found. These heuristics introduce no additional learning parameters and serve only to reconstruct valid MSP-style trees from token-level

predictions. They are applied uniformly across all models and ablations, and therefore do not affect comparative conclusions.

Finally, the reconstructed output is evaluated using the official MSP metrics, which compute MSLAS, LAS, and morphological F1 on content tokens while incorporating abstract-node predictions through their layered dependency relations and morphological features.

5 Experimental Setup

To evaluate whether the proposed joint and typology-aware modeling decisions translate into empirical gains, we conduct experiments on the UniDive 2025 MSP shared task dataset.

Dataset We use the UniDive 2025 Morphosyntactic Parsing (MSP) shared task dataset, covering nine languages: Czech, English, Hebrew, Italian, Polish, Portuguese, Serbian, Swedish, and Turkish. The data is derived from Universal Dependencies treebanks and converted into the MSP format, where dependency structure is defined over content words, while function words, morphology, and unrealized arguments are encoded as layered features and abstract annotations. Each language includes training, development, and test splits following standard UD partitions. Since prediction operates on covered sentences, we tokenize raw text with external tokenizers to match the MSP/UD segmentation. We use Stanza (Qi et al., 2020) for most languages, and UDPipe (Straka and Straková, 2017) for Turkish and Italian.

Baselines and Ablations Traditional UD parsers are not included as baselines, since their formulation assumes all syntactic elements appear as explicit tree nodes and cannot represent MSP's abstract-node projections or morphological layering. As our primary baseline, we use the official shared-task leader system, which implements a joint word-type, dependency parsing, and morphological tagging architecture tailored to MSP but is trained independently for each language. Our ablation analysis systematically evaluates the contribution of typological information in the input representation, typology-conditioned contextual adapters, and the abstract-node modeling module by selectively disabling these components while keeping the remaining architecture fixed.

Training Details All models are trained using the Flair framework with `XLM-RoBERTa-large` as the shared encoder, fine-tuned end-to-end. Experiments are conducted in both single-language and multilingual settings; in the latter, training data is provided via language manifests, enabling joint optimization across MSP languages. Models are trained for 20 epochs using AdamW with a learning rate of 2×10^{-5}, mini-batch size 16, gradient accumulation via chunking, and dropout set to 0.33. Early stopping with learning rate annealing (factor 0.5) is applied, and the best model is selected based on development-set LAS.

Losses for parsing, morphology, and word-type prediction are combined with tuned scalar weights, selected on the development set; in our final configuration, $\lambda_{\text{par}} = 2.0$, $\lambda_{\text{morph}} = 2.0$, and $\lambda_{\text{wt}} = 1.5$. Abstract-node supervision follows a single-stage ramp-up schedule, gradually increasing its weight from 0.01 to 0.5 using a power-based schedule ($\gamma = 2.0$), which delays strong abstract supervision until shared representations stabilize. Typological information is incorporated via 64-dimensional URIEL vectors unless disabled, and contextual adapters are enabled or ablated depending on the experimental configuration.

Evaluation Metrics We follow the UniDive 2025 MSP evaluation protocol. Dependency structure is evaluated using Labeled Attachment Score (LAS) over content tokens. MSLAS extends LAS by requiring the correct prediction of morphological feature bundles attached to content tokens, thereby jointly assessing syntax and morphology. Morphological tagging performance is reported as micro-averaged F1 over feature–value pairs. Word-type classification is evaluated using token-level accuracy, as it controls participation in parsing and tagging. Abstract-node modeling is reflected implicitly in MSLAS and morphological F1 through the correctness of layered dependency relations and morphological features associated with base tokens.

6 Results

We report macro-averaged results and key comparative findings in the main paper. Detailed language-level results for all analyses are provided in the Appendix, beginning with Appendix B.

Main Results Table 1 reports macro-averaged results across all languages. The proposed model consistently outperforms the baseline on both DEV and TEST splits across MSLAS, LAS, and FEATS,

with gains of up to +3.77 MSLAS/LAS and +2.34 FEATS on DEV, and +3.35 LAS and +1.78 FEATS on TEST. While absolute TEST scores are lower than DEV, improvements remain stable across languages, as shown in Appendix B. These results indicate that the proposed joint, typology-aware architecture generalizes beyond development conditions and remains effective under test-time distributional shifts.

Metric	baseline_system	proposed_system	Diff
MSLAS	81.19	**84.42**	↑3.23
LAS	82.62	**85.97**	↑3.35
FEATS	92.02	**93.80**	↑1.78

(a) TEST split (macro averages across languages).

Metric	baseline_system	proposed_system	Diff
MSLAS	81.58	**85.35**	↑3.77
LAS	83.08	**86.85**	↑3.77
FEATS	92.15	**94.49**	↑2.34

(b) DEV split (macro averages across languages).

Table 1: Overall macro-averaged results across languages comparing the baseline system and the proposed model on DEV and TEST splits.

Ablation Analysis The ablation study evaluates how typological information and contextual adapters contribute to multilingual MSP performance: as shown in Table 2, adding typology alone (*multiTypo*) yields marginal or inconsistent changes relative to the base multilingual model, indicating that static typological features are insufficient to bias token-level representations without contextual modulation, whereas adding adapters alone (*multiAdp*) produces consistent gains in MSLAS and LAS, and their combination in the full model (*multiAdpTypo*) achieves the strongest macro-averaged performance across metrics, a pattern that holds across languages in Appendix C.

Internalization of Typology Table 3 compares static URIEL typology with the learned language representation h_{lang} using four alignment metrics that capture global, local, and task-specific structure. *Global RSA vs Data Structure* measures the Spearman correlation between pairwise language distances in the representation space and distances induced by aggregate data-derived morphological features, reflecting overall geometric agreement. *Nearest-Neighbor Match (k=1)* evaluates local consistency by measuring how often a language's closest neighbor in the representation space matches its closest neighbor under data-induced distances. *Morphological Feature Alignment* quantifies how

Metric	multi	multiTypo	multiAdp	multiAdpTypo
MSLAS	84.25	83.96	84.39	**84.42**
LAS	85.79	85.71	85.84	**85.97**
FEATS	93.71	93.55	93.79	**93.80**

(a) TEST split (macro averages).

Metric	multi	multiTypo	multiAdp	multiAdpTypo
MSLAS	85.12	84.98	85.33	**85.35**
LAS	86.67	86.55	**86.89**	86.85
FEATS	94.41	94.35	94.41	**94.49**

(b) DEV split (macro averages).

Table 2: Ablation study with macro-averaged MSLAS, LAS, and FEATS. *multi* is the multilingual joint model without typology or adapters. *multiTypo* adds typology conditioning only. *multiAdp* adds contextual adapters only. *multiAdpTypo* is the full model with typology-conditioned adapters.

well representation distances correlate with cross-lingual differences in pooled morphological feature distributions, while *Dependency Relation Alignment* measures the same correspondence for dependency relation distributions. Across all four metrics, h_{lang} outperforms URIEL, with gains in morphological and dependency alignment, suggesting that typological information is not merely preserved but internalized and reshaped through task-driven learning. Detailed pairwise distance changes and nearest-neighbor shifts are reported in Appendix D. Variants that internalize typology more strongly also yield higher MSLAS/LAS, suggesting that typology internalization is functionally relevant rather than purely representational.

Metric	URIEL	h_{lang}
Global RSA vs Data Structure	0.01	**0.10**
Nearest-Neighbor Match (k=1)	0.00	**0.22**
Morphological Feature Alignment	0.07	**0.48**
Dependency Relation Alignment	-0.27	**0.40**

Table 3: Comparison of URIEL and the learned language representation h_{lang} using the most informative global, local, and task-relevant alignment metrics.

Abstract Node Analysis Table 4 quantifies the effect of explicit abstract node modeling. Enabling abstract modeling improves all metrics on both splits, with particularly large gains in FEATS (+2.76 on TEST, +3.52 on DEV), indicating improved recovery of morphosyntactic information. At the language level (Appendix Table E), gains vary substantially: languages with frequent abstract-node phenomena show consistent improvements, while languages with sparse abstract annotations exhibit small or neutral changes. This variability motivates a more detailed analysis

of abstract-node distribution and language-specific behavior in the following section.

Metric	without_abstracts	with_abstracts	Diff
MSLAS	82.61	**84.42**	↑1.81
LAS	84.63	**85.97**	↑1.34
FEATS	91.04	**93.80**	↑2.76

(a) TEST split (macro averages across languages).

Metric	without_abstracts	with_abstracts	Diff
MSLAS	82.21	**85.35**	↑3.14
LAS	84.14	**86.85**	↑2.71
FEATS	90.97	**94.49**	↑3.52

(b) DEV split (macro averages across languages).

Table 4: Macro-averaged DEV and TEST results across languages comparing models with and without explicit abstract node modeling for MSLAS, LAS, and FEATS.

7 Discussion

This section gives details about task interaction, errors, comparative insights.

Interaction Between Tasks Our model improves MSLAS, LAS, and FEATS simultaneously on both splits (+3.23/+3.35/+1.78 on TEST; +3.77/+3.77/+2.34 on DEV; Table 1), suggesting that the shared content-only representation benefits both dependency structure and morphosyntactic labeling. The ablations further show that typology is most useful when it modulates contextualization: typology-only (*multiTypo*) is inconsistent, adapters-only (*multiAdp*) is consistently better, and combining them yields the strongest macro results (*multiAdpTypo*; Table 2), consistent with the stronger data-alignment of h_{lang} over URIEL (Table 3).

Error Analysis The smaller improvements on TEST relative to DEV in the abstract modeling comparison (+1.81/+1.34/+2.76 vs. +3.14/+2.71/+3.52; Table 4) are consistent with test-time sparsity and imbalance of abstract phenomena. Prior MSP analysis (Acar and Eryiğit, 2025) shows that abstract nodes are extremely rare in TEST for English and Serbian (0.39% each), whereas languages like Turkish have much higher rates (13.45%). In such settings, abstract-node supervision and evaluation contribute only a weak signal, naturally limiting macro-averaged gains on TEST. This explains the near-zero deltas for EN/S-R/PT in Appendix E and why they can depress macro TEST scores, motivating their exclusion from fine-grained abstract-focused comparison.

Linguistic and Typological Insights The per-language effects match typological expectations and align with our internalization analysis. Typology-conditioned adapters (*multiAdpTypo*) remain competitive across diverse languages (Appendix Table 6), supporting the idea that typology helps most when it modulates contextualization rather than acting as a static prior. The *Internalization of Typology* results further show that the learned language representation h_{lang} is better aligned with data-driven morphosyntactic variation than URIEL (Table 3), with gains on morphology- and dependency-sensitive alignments, indicating a task-shaped reorganization of cross-lingual geometry. Consistently, explicit abstract-node modeling yields its benefits in languages where unrealized arguments contribute layered agreement and argument-structure signals: Turkish shows the largest gains (e.g., +10.00 MSLAS / +10.08 LAS on TEST; Table 8), with consistent improvements also in Polish and Czech, while languages with sparse abstract nodes show limited change.

8 Conclusion

We introduced a multilingual, typology-aware joint model for MSP that unifies dependency parsing, morphological tagging, word-type prediction, and explicit abstract-node modeling in a single architecture. By directly modeling unrealized arguments and morphological layering, the proposed system produces structurally valid MSP outputs beyond the capabilities of traditional UD parsers.

Across DEV and TEST splits, the model outperforms the shared-task system, with gains of up to +3.77 MSLAS/LAS and +2.34 FEATS. Ablation results show that typological information is effective when it modulates contextual representations via adapters, rather than acting as static input. Explicit abstract-node modeling yields the improvements in morphologically rich languages where unrealized arguments are frequent, while its impact is limited in languages with sparse abstract phenomena.

These results show that MSP, a formalism with non-overt syntactic material, cannot be addressed by token-centric parsing architectures alone. By modeling unrealized arguments as projections and conditioning representations on typological signals, the proposed framework outlines a template for extending parsing models to settings where syntactic structure and morphological realization diverge from surface form, and motivates future work on fully learnable abstract-node decoding and deeper typology integration in multilingual parsing.

Limitations

While the proposed system advances the state of the art in multilingual morphosyntactic parsing, several limitations remain.

First, abstract nodes are inherently sparse and unevenly distributed across languages and splits, which limits the robustness of abstract-node learning, particularly in low-frequency settings. Although our joint formulation improves abstract-node recovery in morphologically rich languages, future work could explore stronger regularization strategies or data augmentation techniques tailored to abstract-node sparsity.

Second, abstract-head prediction currently relies on deterministic reconstruction heuristics, motivated by the fact that many abstract labels are structurally trivial in the training data. While effective in practice, this design constrains the model's capacity to learn abstract structure end-to-end. Developing fully learnable abstract-head and attachment mechanisms, potentially integrated with structured inference, remains an important direction.

Third, while typology-aware conditioning improves multilingual generalization, our analysis treats typology as a unified signal. A more fine-grained investigation of feature-specific typological effects, such as isolating which morphosyntactic properties contribute most to gains in parsing, morphology, or abstract modeling, would provide deeper linguistic insight.

Fourth, typological information is limited to URIEL-style features and learned language embeddings. Extending typology awareness with additional resources, such as language-specific morphological inventories, diachronic features, or learned typological representations from large multilingual models, may further improve adaptability.

Finally, while we performed hyperparameter tuning for key components, the explored search ranges and resolution were limited by computational constraints. A more extensive sweep (e.g., broader ranges, finer granularity, and more interactions across loss weights, adapter capacity, and abstract supervision schedules), as well as explicit integration of tokenization choices into the learning objective, could yield further improvements, especially for morphologically complex languages.

References

Kutay Acar and Gülşen Eryiğit. 2025. Typology-aware multilingual morphosyntactic parsing with functional node filtering. In *Proceedings of The UniDive 2025 Shared Task on Multilingual Morpho-Syntactic Parsing*, pages 27–33, Ljubljana, Slovenia. Association for Computational Linguistics.

Demian Inostroza Améstica, Meladel Mistica, Ekaterina Vylomova, Chris Guest, and Kemal Kurniawan. 2025. A joint multitask model for morpho-syntactic parsing. In *Proceedings of The UniDive 2025 Shared Task on Multilingual Morpho-Syntactic Parsing*, pages 19–26, Ljubljana, Slovenia. Association for Computational Linguistics.

Guntis Bārzdiņš, Normunds Grūzītis, Gunta Nešpore, and Baiba Saulīte. 2007. Dependency-based hybrid model of syntactic analysis for the languages with a rather free word order. In *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA 2007)*, pages 13–20, Tartu, Estonia. University of Tartu, Estonia.

Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal Dependencies. *Computational Linguistics*, 47(2):255–308.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Omer Goldman and Reut Tsarfaty. 2022. Morphology without borders: Clause-level morphology. *Transactions of the Association for Computational Linguistics*, 10:1455–1472.

Omer Goldman, Leonie Weissweiler, Kutay Acar, Diego Alves, Anna Baczkowska, Gülşen Eryiğit, Lenka Krippnerová, Adriana Pagano, Tanja Samardžić, Luigi Talamo, Alina Wróblewska, Daniel Zeman, Joakim Nivre, and Reut Tsarfay. 2025. Findings of the UniDive 2025 shared task on multilingual morpho-syntactic parsing. In *Proceedings of The UniDive 2025 Shared Task on Multilingual Morpho-Syntactic Parsing*, pages 1–18, Ljubljana, Slovenia. Association for Computational Linguistics.

Johan Hall, Jens Nilsson, Joakim Nivre, Gülşen Eryiğit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? a study in multilingual parser optimization. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 933–939, Prague, Czech Republic. Association for Computational Linguistics.

Dan Kondratyuk and Milan Straka. 2019. 75 languages, 1 model: Parsing Universal Dependencies universally. 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

2779–2795, Hong Kong, China. Association for Computational Linguistics.

Patrick Littell, David R. Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. Uriel and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 8–14. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 149–160, Nancy, France.

Joakim Nivre, Ali Basirat, Luise Dürlich, and Adam Moss. 2022. Nucleus composition in transition-based dependency parsing. *Computational Linguistics*, 48(4):849–886.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 221–225, New York City. Association for Computational Linguistics.

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.

Agata Savary, Daniel Zeman, Verginica Barbu Mititelu, Anabela Barreiro, Olesea Caftanatov, Marie-Catherine de Marneffe, Kaja Dobrovoljc, Gülşen Eryiğit, Voula Giouli, Bruno Guillaume, Stella Markantonatou, Nurit Melnik, Joakim Nivre, Atul Kr. Ojha, Carlos Ramisch, Abigail Walsh, Beata Wójtowicz, and Alina Wróblewska. 2024. UniDive: A COST action on universality, diversity and idiosyncrasy in language technology. In *Proceedings of the 3rd Annual Meeting of the Special Interest Group on Underresourced Languages @ LREC-COLING 2024*, pages 372–382, Torino, Italia. ELRA and ICCL.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2020. UDapter: Language adaptation for truly Universal Dependency parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2302–2315, Online. Association for Computational Linguistics.

Ahmet Üstün, Arianna Bisazza, Gosse Bouma, and Gertjan van Noord. 2022. UDapter: Typology-based language adapters for multilingual dependency parsing and sequence labeling. *Computational Linguistics*, 48(3):555–592.

A System Architecture Diagram

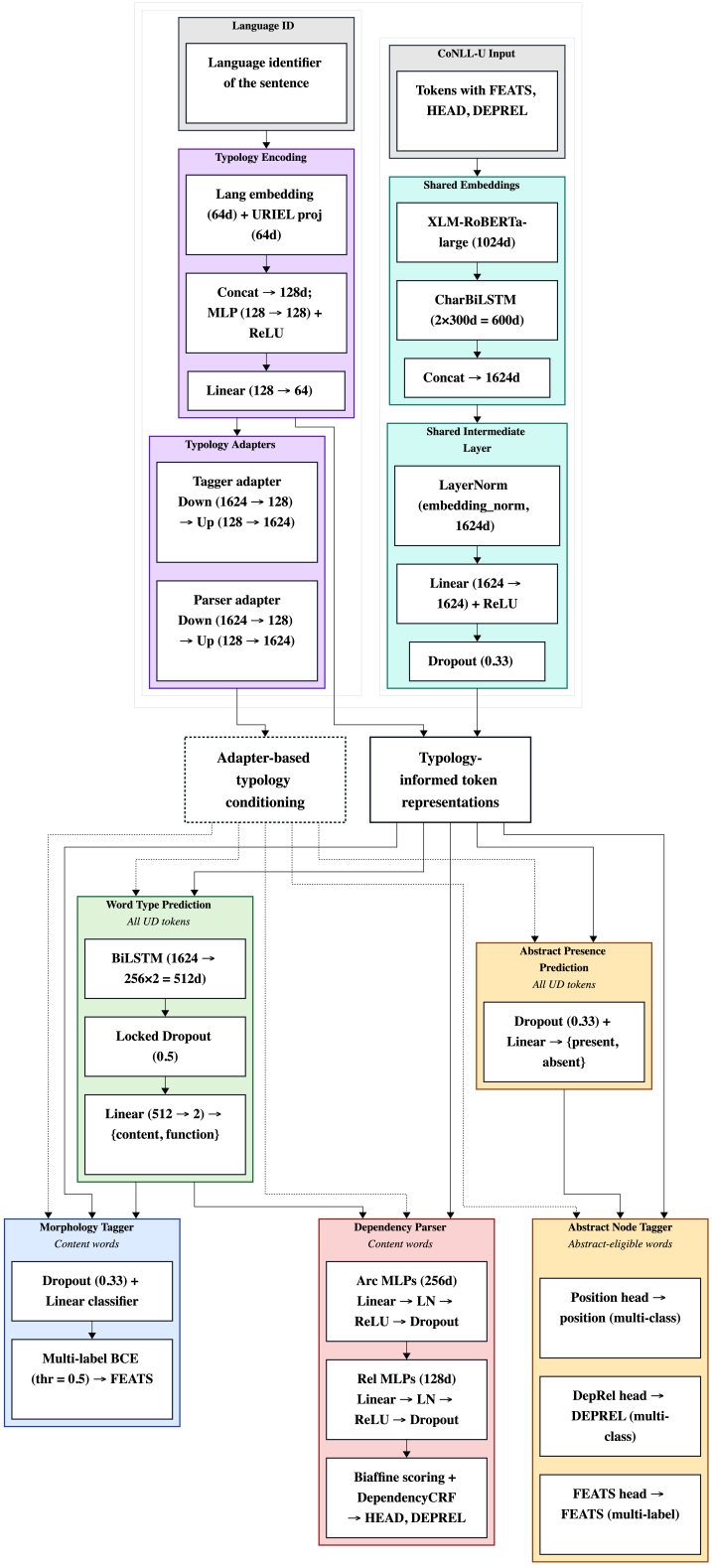

Figure 2: Architecture of the proposed typology-aware joint MSP model. Light-blue blocks denote shared text encoding, while pink blocks represent typology processing, influencing task heads via dotted connections and input representations via solid connections. The green word type predictor gates the morphology (blue) and dependency parsing (red) heads to operate on content words only. Yellow blocks model abstract nodes through a separate gating mechanism and three heads for position, dependency relation, and morphological features.

B Main Results

Table 5: Performance per language using covered CoNLL-U. Each subtable reports MSLAS, LAS, and FEATS scores. (Scores might slightly differ after hyperparameter tuning is completed.)

(a) Test Split

MSLAS	**AVG**	cz	en	he	it	pl	pt	sr	sv	tr
baseline	81.19	87.12	83.79	68.71	84.53	85.69	88.87	86.61	86.64	58.74
our_model	**84.42**	**88.97**	**84.75**	**72.42**	**87.21**	**89.86**	**89.48**	**87.55**	**89.68**	**69.90**
Diff	↑3.23	↑1.85	↑0.96	↑3.71	↑2.68	↑4.17	↑0.61	↑0.94	↑3.04	↑11.16

LAS	**AVG**	cz	en	he	it	pl	pt	sr	sv	tr
baseline	82.62	88.00	85.08	71.45	85.38	87.23	89.55	88.27	87.67	60.93
our_model	**85.97**	**89.94**	**86.28**	**75.63**	**87.97**	**91.47**	**90.17**	**89.18**	**90.71**	**72.34**
Diff	↑3.35	↑1.94	↑1.20	↑4.18	↑2.59	↑4.24	↑0.62	↑0.91	↑3.04	↑11.41

FEATS	**AVG**	cz	en	he	it	pl	pt	sr	sv	tr
baseline	92.02	95.16	**94.90**	83.45	93.12	93.50	**94.83**	**95.58**	95.58	82.07
our_model	**93.80**	**97.05**	94.52	**84.96**	**94.70**	**96.54**	94.79	95.19	**96.17**	**90.28**
Diff	↑1.78	↑1.89	↓0.38	↑1.51	↑1.58	↑3.04	↓0.04	↓0.39	↑0.59	↑8.21

(b) Development Split

MSLAS	**AVG**	cz	en	he	it	pl	pt	sr	sv	tr
baseline	81.58	87.54	85.39	74.98	85.96	85.60	88.92	85.64	85.31	54.91
our_model	**85.35**	**89.32**	**87.79**	**78.23**	**88.48**	**89.38**	**92.16**	**87.67**	**87.20**	**67.89**
Diff	↑3.77	↑1.78	↑2.40	↑3.25	↑2.52	↑3.78	↑3.24	↑2.03	↑1.89	↑12.98

LAS	**AVG**	cz	en	he	it	pl	pt	sr	sv	tr
baseline	83.08	88.41	86.89	77.58	86.81	87.23	89.72	87.48	86.24	57.35
our_model	**86.85**	**90.18**	**89.30**	**80.91**	**89.36**	**90.99**	**92.97**	**89.45**	**88.32**	**70.21**
Diff	↑3.77	↑1.77	↑2.41	↑3.33	↑2.55	↑3.76	↑3.25	↑1.97	↑2.08	↑12.86

FEATS	**AVG**	cz	en	he	it	pl	pt	sr	sv	tr
baseline	92.15	95.43	94.68	86.70	93.44	93.50	94.52	94.48	**95.44**	81.17
our_model	**94.49**	**97.17**	**95.35**	**88.70**	**95.34**	**96.32**	**97.01**	**95.03**	95.22	**90.28**
Diff	↑2.34	↑1.74	↑0.67	↑2.00	↑1.90	↑2.82	↑2.49	↑0.55	↓0.22	↑9.11

C Ablation Analysis

Table 6: Detailed ablation results (MSLAS/LAS/FEATS). For each language and metric, the best variant is shown in **bold**.

(a) TEST split

Lang	MULTI			MULTITYPO			MULTIADP			MULTIADPTYPO		
	MSLAS	LAS	FEATS	MSLAS	LAS	FEATS	MSLAS	LAS	FEATS	MSLAS	LAS	FEATS
cz	88.92	89.95	96.64	88.38	89.43	96.80	**89.21**	**90.19**	**97.13**	88.97	89.94	97.05
en	83.93	85.76	94.01	83.82	85.67	94.02	**85.18**	**86.72**	94.35	84.75	86.28	**94.52**
he	**72.72**	75.61	**85.44**	71.58	74.30	85.26	72.61	75.45	85.42	72.42	**75.63**	84.96
it	87.09	87.79	94.53	**87.38**	**88.10**	**94.84**	87.03	87.71	94.62	87.21	87.97	94.70
pl	89.67	91.32	96.36	89.58	91.17	96.41	89.78	91.33	96.42	**89.86**	**91.47**	**96.54**
pt	89.28	90.00	94.79	88.85	89.62	94.39	89.31	90.02	94.70	**89.48**	**90.17**	**94.79**
sr	87.52	89.03	95.18	**87.65**	**89.24**	**95.39**	87.07	88.54	95.13	87.55	89.18	95.19
sv	89.65	90.60	**96.28**	88.98	90.00	95.98	89.37	90.38	96.16	**89.68**	**90.71**	96.17
tr	69.45	72.03	90.15	69.23	71.87	89.91	**69.95**	72.23	90.14	69.90	**72.34**	**90.28**
AVG	84.25	85.79	93.71	83.96	85.71	93.55	84.39	85.84	93.79	**84.42**	**85.97**	**93.80**

(b) DEV split

Lang	MULTI			MULTITYPO			MULTIADP			MULTIADPTYPO		
	MSLAS	LAS	FEATS	MSLAS	LAS	FEATS	MSLAS	LAS	FEATS	MSLAS	LAS	FEATS
cz	**89.77**	**90.65**	97.21	89.16	90.10	96.99	89.73	90.60	**97.26**	89.32	90.18	97.17
en	87.40	89.01	95.20	87.52	89.02	95.24	87.59	89.26	95.15	**87.79**	**89.30**	**95.35**
he	78.12	**80.94**	88.67	77.04	79.82	88.24	77.61	80.39	88.30	**78.23**	80.91	**88.70**
it	88.20	89.02	95.23	**88.67**	**89.57**	**95.50**	88.66	89.51	95.43	88.48	89.36	95.34
pl	88.89	90.61	96.06	88.72	90.34	96.11	89.20	90.86	95.99	**89.38**	**90.99**	**96.32**
pt	92.49	93.22	**97.32**	91.61	92.55	96.79	**92.58**	**93.39**	97.16	92.16	92.97	97.01
sr	87.04	88.91	94.85	85.96	87.86	94.24	86.87	88.65	94.80	**87.67**	**89.45**	**95.03**
sv	86.88	87.93	95.00	86.99	88.15	95.31	**87.51**	**88.76**	**95.49**	87.20	88.32	95.22
tr	67.25	69.78	90.19	67.02	69.70	89.58	**68.24**	**70.63**	90.11	67.89	70.21	**90.28**
AVG	85.12	86.67	94.41	84.98	86.55	94.35	85.33	**86.89**	94.41	**85.35**	86.85	**94.49**

D Internalization of Typology

Table 7: Pairwise typological reorganization from URIEL to the learned language representation h_{lang}. Subtable (a) shows the most strongly converging language pairs, reporting distances under both representations and their change. Subtable (b) reports nearest-neighbor shifts, with 8 out of 9 languages changing their closest neighbor.

(a) Strongest converging language pairs from URIEL to h_{lang} (largest negative distance change Δ).

Lang$_1$	**Lang$_2$**	**URIEL dist.**	**h_{lang} dist.**	Δ
he	pt	1.36	0.63	-0.74
it	pt	1.25	0.54	-0.72
pl	sv	1.26	0.56	-0.70
pt	sv	1.29	0.60	-0.70
cs	pl	1.08	0.42	-0.66
sr	sv	1.20	0.58	-0.62
en	sr	1.18	0.56	-0.62
en	pt	1.11	0.49	-0.62
en	pl	1.01	0.43	-0.58
cs	sr	1.02	0.45	-0.57
cs	pt	1.05	0.50	-0.56
he	sv	1.24	0.69	-0.55

(b) Nearest-neighbor changes ($k = 1$) from URIEL to h_{lang}.

Language	**URIEL NN**	**h_{lang} NN**	**Changed**
cs	en	pl	Yes
en	cs	cs	No
he	pl	sr	Yes
it	tr	pt	Yes
pl	sr	cs	Yes
pt	tr	en	Yes
sr	pl	cs	Yes
sv	en	cs	Yes
tr	it	en	Yes

E Abstract Node Analysis

Table 8: Effect of abstract node modeling. Baseline corresponds to NO_ABSTRACT. Each subtable reports MSLAS, LAS, and FEATS scores.

(a) Test Split

MSLAS	**AVG**	cz	en	he	it	pl	pt	sr	sv	tr
without_abs	82.61	86.82	84.75	70.54	86.06	86.75	89.48	87.55	89.68	59.90
with_abstracts	**84.42**	**88.97**	84.75	**72.42**	**87.21**	**89.86**	89.48	87.55	89.68	**69.90**
Diff	↑1.81	↑2.15	0.00	↑1.88	↑1.15	↑3.11	0.00	0.00	0.00	↑10.00

LAS	**AVG**	cz	en	he	it	pl	pt	sr	sv	tr
without_abs	84.63	87.78	86.28	73.75	86.83	88.40	90.17	89.18	90.70	62.26
with_abstracts	**85.97**	**89.94**	86.28	**75.63**	**87.97**	**91.47**	90.17	89.18	**90.71**	**72.34**
Diff	↑1.34	↑2.16	0.00	↑1.88	↑1.14	↑3.07	0.00	0.00	↑0.01	↑10.08

FEATS	**AVG**	cz	en	he	it	pl	pt	sr	sv	tr
without_abs	91.04	95.08	94.52	83.36	93.69	93.61	94.79	95.19	96.15	82.14
with_abstracts	**93.80**	**97.05**	94.52	**84.96**	**94.70**	**96.54**	94.79	95.19	**96.17**	**90.28**
Diff	↑2.76	↑1.97	0.00	↑1.60	↑1.01	↑2.93	0.00	0.00	↑0.02	↑8.14

(b) Development Split

MSLAS	**AVG**	cz	en	he	it	pl	pt	sr	sv	tr
without_abs	82.21	87.61	87.62	76.37	87.09	86.57	89.58	87.36	87.19	57.50
with_abstracts	**85.35**	**89.32**	**87.79**	**78.23**	**88.48**	**89.38**	**92.16**	**87.67**	**87.20**	**67.89**
Diff	↑3.14	↑1.71	↑0.17	↑1.86	↑1.39	↑2.81	↑2.58	↑0.31	↑0.01	↑10.39

LAS	**AVG**	cz	en	he	it	pl	pt	sr	sv	tr
without_abs	84.14	88.47	89.12	78.93	87.98	88.21	90.36	89.13	88.30	59.75
with_abstracts	**86.85**	**90.18**	**89.30**	**80.91**	**89.36**	**90.99**	**92.97**	**89.45**	**88.32**	**70.21**
Diff	↑2.71	↑1.71	↑0.18	↑1.98	↑1.38	↑2.78	↑2.61	↑0.32	↑0.02	↑10.46

FEATS	**AVG**	cz	en	he	it	pl	pt	sr	sv	tr
without_abs	90.97	95.39	95.20	87.12	94.05	93.71	94.56	94.73	95.21	81.98
with_abstracts	**94.49**	**97.17**	**95.35**	**88.70**	**95.34**	**96.32**	**97.01**	**95.03**	**95.22**	**90.28**
Diff	↑3.52	↑1.78	↑0.15	↑1.58	↑1.29	↑2.61	↑2.45	↑0.30	↑0.01	↑8.30