applied sciences

MDPI

Article

SAC-XLM: Structure Discovery Cross-Lingual Transfer Through Advantage Actor–Critic Algorithm for Multilingual Model

Yijing Hai, Gyu-Min Park and Seong-Bae Park *

School of Computing, Kyung-Hee University, Giheung-gu, Yongin-si 17104, Gyeonggi-do, Republic of Korea; roselyn@khu.ac.kr (Y.H.); pgm1219@khu.ac.kr (G.-M.P.)
* Correspondence: sbpark71@khu.ac.kr

Abstract

Multilingual language models have attracted increasing attention as language modeling has advanced. However, the uneven availability of high-quality labeled data across languages can lead to disparities in downstream performance. Struct-XLM addresses this challenge by using reinforcement learning to discover shared syntactic structures, reducing its reliance on labeled corpora. However, its use of REINFORCE can produce high-variance policy gradient estimates, resulting in unstable training. This paper proposes SAC-XLM, an optimization framework for multilingual structure discovery based on advantage actor–critic (A2C) learning. SAC-XLM combines a pretrained structural encoder, an actor network, a critic network, and a reward acquirer to learn structural representations across languages. The critic network provides a state-dependent value baseline for advantage estimation, while the reward acquirer evaluates the contribution of structural actions to cross-lingual alignment. Together, these components support more stable policy optimization. Experiments on seven XTREME tasks show that SAC-XLM improves training stability and efficiency compared with Struct-XLM. It reduces pretraining time to approximately 40% of that required by Struct-XLM while preserving overall downstream performance.

Keywords: multilingual language models; cross-lingual models; reinforcement learning; advantage actor–critic

1. Introduction

Cross-lingual representation learning is an important area of multilingual natural language processing because it enables knowledge transfer across languages and reduces reliance on language-specific annotated resources. Google introduced mBERT [1], based on BERT [2], which supports 104 languages and is pretrained on Wikipedia corpora using masked language modeling (MLM) and next sentence prediction (NSP). It was the first large language model (LLM) to demonstrate the feasibility of zero-shot cross-lingual transfer. Subsequently, XLM [3] introduced MLM and translation language modeling (TLM) objectives and adopted a shared subword vocabulary. This significantly improved multilingual transfer, including performance on low-resource languages. XLM-R (XLM-RoBERTa) [4] further improved multilingual performance through pretraining on 2.5 TB of CC-100 corpora covering 100 languages, using SentencePiece [5] to construct a shared subword vocabulary. Despite these advances, such models still rely heavily on large amounts of annotated training data. Collecting such data is costly, particularly for low-resource languages. Differences in the availability and quality of annotations across languages lead to substantial performance disparities.

Check for updates

Academic Editors: Paolo Renna and Andrea Prati

Received: 8 August 2026
Revised: 21 September 2026
Accepted: 22 October 2026
Published: 25 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

To address this issue, Wu and Lu proposed Struct-XLM [6], which learns multilingual representations from limited data by discovering latent constituent structures. Structural information can improve representation quality under limited supervision [7,8]. Struct-XLM uses reinforcement learning (RL) to guide structure prediction through reward signals, allowing such information to be learned automatically.

A key limitation of Struct-XLM is training instability. Its REINFORCE algorithm [9] estimates policy gradients from sampled rewards using Monte Carlo sampling [10]. These estimates can have high variance, causing large fluctuations in parameter updates and slowing convergence.

Several approaches have been proposed to mitigate training instability. Potential-based reward shaping (PBRS) [11] dynamically modifies rewards according to the estimated progress of learning. By providing additional rewards for beneficial decisions and penalties for less desirable ones, it offers more informative feedback for optimization. Proximal policy optimization (PPO) [12] improves stability by constraining the magnitude of policy updates, limiting the deviation between successive policies. Advantage actor–critic (A2C) [13] uses an actor–critic architecture in which a state-dependent value function estimates the value of each state. Advantages are computed by comparing observed rewards with these estimates, reducing the variance of the signals used to guide model updates.

This paper proposes the Structure discovery Actor–Critic Cross-lingual Language Model (SAC-XLM), an A2C-based optimization framework for multilingual structure discovery. To integrate A2C with multilingual pretrained models, SAC-XLM uses a dual-network architecture with shared masked self-attention and linear layers. The critic network estimates state values, which are compared with observed rewards to obtain advantage estimates. This value-based estimation provides more stable lower-variance signals for policy optimization. A structure-aware reward is also designed to alleviate sparse feedback. By comparing the semantic similarity of bilingual sentence representations with and without structural information, this reward guides both the direction and magnitude of model updates. Together, these components reduce training instability and accelerate convergence, enabling more effective structure learning under limited supervision.

The choice of A2C is motivated by the requirements of multilingual structure discovery, which involves continuous exploration of diverse latent structures across languages. A2C reduces policy gradient variance through value-based advantage estimation, improving training stability without explicitly restricting policy exploration. PBRS relies on task-specific reward engineering, while PPO constrains policy updates and thereby limits the flexibility of exploration. A2C offers a more suitable balance between training stability and exploration flexibility for multilingual structure discovery.

The main contributions of this paper are as follows:

- This paper proposes SAC-XLM, an A2C-based framework for stable multilingual structure learning. Its actor–critic architecture enables value-guided policy optimization under limited supervision.
- A structure-aware reward is designed for the A2C framework that evaluates structural actions by comparing bilingual sentence representations generated with and without the actor network's actions. This reward provides a clear optimization direction and encodes the magnitude of updates.
- Experiments on seven XTREME tasks [14] show improved training stability and efficiency while maintaining comparable overall multilingual performance. SAC-XLM requires approximately 40% of Struct-XLM's pretraining time and 73% of its average downstream fine-tuning time.

2. Related Work

2.1. Multilingual Language Models

Multilingual natural language processing (NLP) has advanced considerably with the development of large-scale pretrained language models [15,16]. Models such as mBART [17] and mT5 [18] have extended language coverage to more than 100 languages using denoising objectives on large multilingual corpora. Despite this broad coverage, substantial performance differences persist across languages. In particular, these models exhibit the *curse of multilinguality* [4], especially for low-resource languages.

Subsequent research has explored several strategies to reduce these cross-lingual performance gaps. The first strategy focuses on improving cross-lingual alignment. For instance, VECO 2.0 [19] incorporates a token-level contrastive learning objective to explicitly align semantically similar words across different languages, thereby facilitating more effective knowledge transfer. Similarly, InfoXLM [20] improves cross-lingual alignment by maximizing the mutual information between parallel sentence representations. Another strategy scales up training data and computational resources to improve multilingual generalization. PaLM [21] uses the Pathways system [22] for training on a corpus of 780 billion tokens. Similarly, XGLM [23] is trained on a 500-billion-token dataset. More recently, mmBERT [24] has revisited encoder-only multilingual pretraining by scaling training to over three trillion tokens covering more than 1800 languages. A third strategy expands the language coverage of training data. NLLB [25] supports translation across 200 languages by incorporating large-scale multilingual parallel data, particularly for underrepresented and low-resource languages. This broader coverage improves generalization across linguistically diverse settings. However, these approaches rely heavily on large-scale parallel or annotated data, which are particularly costly to obtain for low-resource languages. The resulting resource disparities contribute to uneven multilingual performance.

Struct-XLM [6] reduces reliance on large-scale parallel data through an RL-based framework for multilingual modeling. It formulates structural alignment as a decision-making process in which RL selects and optimizes structural actions. Using a parallel corpus of only 8 MB, it demonstrates the potential of structure learning under limited supervision. However, its use of vanilla REINFORCE can produce high-variance gradient estimates, leading to unstable training and slow convergence.

Representation alignment has also been investigated beyond cross-lingual language modeling. For example, NoCo [26] improves multimodal entity alignment across heterogeneous knowledge graphs through noise augmentation and confidence-based dynamic fusion, highlighting the importance of robustly aligning representations from different information sources. In another application of language model adaptation, GeoTP [27] uses instruction tuning to enable large language models to plan parameterized GIS tool-use chains. Although its objective differs from cross-lingual representation learning, it illustrates how a tailored training strategy can adapt a language model to a specific task.

2.2. Reinforcement Learning

Reinforcement learning optimizes decision-making policies through interaction with an environment. REINFORCE [9] is a foundational policy-gradient method that uses sampled trajectories to estimate gradients of the expected return. Its reliance on Monte Carlo sampling [10] can lead to high-variance gradient estimates, causing oscillations during training and slowing convergence. These limitations can also affect models that use REINFORCE for multilingual structure learning.

PPO [12] improves training stability by constraining the magnitude of policy changes at each update. This mechanism prevents excessively large updates but also limits the flexibility of policy adaptation, particularly in tasks involving discrete structured decisions that require sufficient exploration of the solution space.

A2C [13] addresses the high variance of REINFORCE through state-dependent value estimation. In its actor–critic architecture, the critic network estimates the expected value of a state. The advantage is then computed as the difference between the observed reward and this estimate. These advantage-based signals guide policy updates, reducing gradient variability and improving training stability and efficiency. The update scheme does not explicitly constrain policy changes, preserving flexibility during exploration.

3. SAC-XLM

3.1. Overview

SAC-XLM is an A2C-based multilingual model with four components: a structural encoder, an actor network, a critic network, and a reward acquirer (Figure 1).

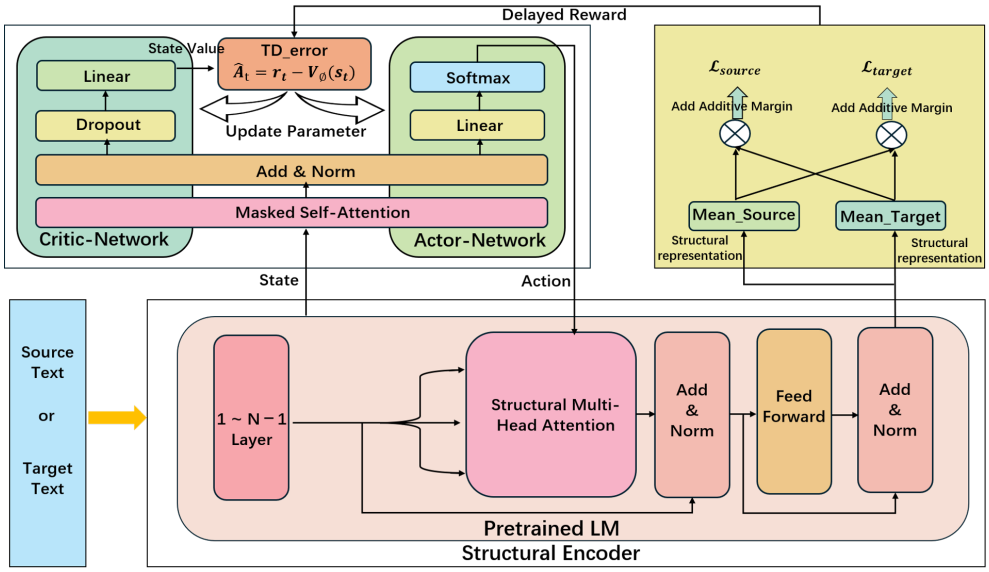

Figure 1. Overview of the proposed SAC-XLM framework.

To enable cross-lingual representation learning, the model is trained on parallel sentence pairs (X, Y), where X and Y denote the source and target language sentences, respectively. Each sentence is represented as a sequence of tokens. The source sentence is denoted as $X = (x_1, x_2, ..., x_T)$ and the target sentence as $Y = (y_1, y_2, ..., y_{T'})$, where T and T' denote the lengths of the source and target sentences, respectively. The model processes these parallel inputs using a Siamese architecture (Figure 2), with two symmetric branches that share parameters across corresponding components. For each sentence pair (X, Y), the lower layers of the structural encoder first produce state sequences. The actor network uses these states to generate structural actions, which are converted into action matrices. These matrices are fed back into the structural encoder to produce a pair of structural representations. The reward acquirer evaluates the representations to obtain reward signals, while the critic network estimates state values. The values and delayed rewards are used to compute temporal difference (TD) errors [28–30] as advantage estimates. These estimates guide updates to the actor network and critic network, completing the optimization loop.

3.2. Structural Encoder

The structural encoder maps an input sentence to a structural representation using a multilingual pretrained language model (PLM). It consists of N stacked layers: the first $N-1$ produce contextual representations, and the final layer incorporates structural information.

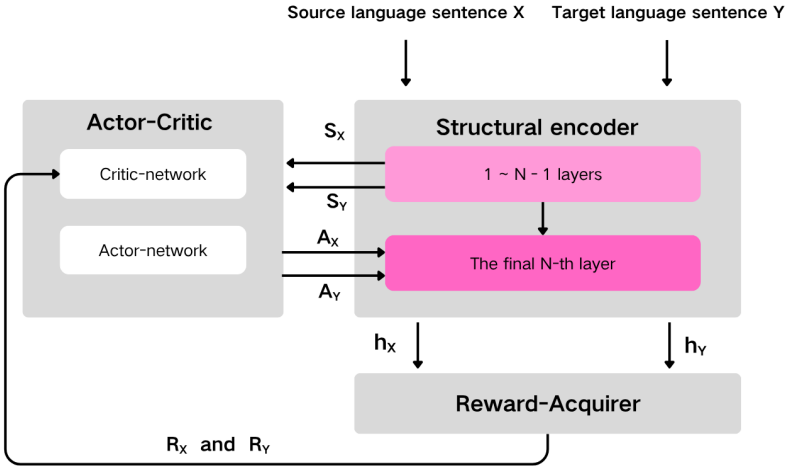

Figure 2. Illustration of the Siamese framework for cross-lingual structural representation learning.

Given a sentence pair (X, Y), the first $N-1$ layers of the encoder transform the source and target sentences independently into corresponding state sequences. Specifically, the source sentence X is mapped to a state sequence $S_X = (s_1^x, s_2^x, ..., s_T^x) \in \mathbb{R}^{T \times d_{model}}$ and the target sentence Y is mapped to $S_Y = (s_1^y, s_2^y, ..., s_{T'}^y) \in \mathbb{R}^{T' \times d_{model}}$, where $s_t \in \mathbb{R}^{d_{model}}$ corresponds to the contextual representation of the t-th token and d_{model} denotes the representation dimension of the encoder. These state sequences capture contextual semantic information and serve as inputs to other components in the framework.

At the final N-th layer, structural information is incorporated into the attention mechanism through two binary action matrices generated by the actor network: $A_X \in \{0,1\}^{T \times T}$ for the source sentence, and $A_Y \in \{0,1\}^{T' \times T'}$ for the target sentence. Each input sentence may contain multiple latent phrases, each defined as a contiguous sequence of tokens. The matrices A_X and A_Y encode these phrase boundaries and use them to modulate attention within the corresponding sentences. Within this final layer, the state representations S_X and S_Y produced by the preceding $N-1$ layers are passed through the query, key, and value linear transformations of the structural multi-head attention module to obtain Q_X, K_X, and V_X for the source branch and Q_Y, K_Y, and V_Y for the target branch, respectively. For each attention head, the query, key, and value matrices and the original attention probability matrices are computed as follows:

$$\begin{aligned} Q_X &= S_X W_{\text{enc}}^Q, \quad K_X = S_X W_{\text{enc}}^K, \quad V_X = S_X W_{\text{enc}}^V, \\ Q_Y &= S_Y W_{\text{enc}}^Q, \quad K_Y = S_Y W_{\text{enc}}^K, \quad V_Y = S_Y W_{\text{enc}}^V, \\ E_0^X &= \text{softmax}\left(\frac{Q_X K_X^\top}{\sqrt{d_k}}\right), \quad E_0^Y = \text{softmax}\left(\frac{Q_Y K_Y^\top}{\sqrt{d_k}}\right), \end{aligned} \tag{1}$$

where W_{enc}^Q, W_{enc}^K, and W_{enc}^V denote the learnable projection matrices that transform the state representations into the query, key, and value spaces, respectively, and d_k denotes the dimension of the query and key vectors.

To incorporate structural information, the action matrices are combined with the attention weights:

$$E^X = E_0^X \odot A_X, \quad E^Y = E_0^Y \odot A_Y, \tag{2}$$

where $\odot$ denotes element-wise multiplication. Through this mechanism, the action matrices inject structural information into the attention process, guiding the model to capture latent structural dependencies. The structural representations h_X and h_Y for each branch are then obtained by applying the modified attention weights to the value matrices:

$$h_X = E^X V_X, \quad h_Y = E^Y V_Y. \tag{3}$$

The resulting representations h_X and h_Y incorporate both contextual semantics and structural information. These computations are performed in parallel across the attention heads in each branch. The head outputs are concatenated to obtain structural representations of dimension $d_{model} = n \times d_k$, where n is the number of heads. Mean pooling over the token representations then produces one sentence-level embedding for each branch.

3.3. Actor Network

The actor network is designed to generate structural actions from input state sequences. Given the state sequence $S = (s_1, s_2, \ldots, s_T) \in \mathbb{R}^{T \times d_{\text{model}}}$ produced by the first $N-1$ layers of the structural encoder, the actor network generates an action sequence $a = (a_1, a_2, \ldots, a_T)$, where S represents the complete sequence and s_t denotes the state representation of its t-th token. For each token, a binary action $a_t \in \{0, 1\}$ is generated: $a_t = 0$ indicates that the token lies within a phrase, while $a_t = 1$ indicates a phrase boundary.

The actor network applies self-attention to the complete state sequence S, producing an attention-enhanced sequence $H = (h_1, h_2, \ldots, h_T)$. Each token-level action depends on contextual information from the full input sequence. The computations are defined as follows:

$$Q = SW^Q_{\text{act}}, \qquad K = SW^K_{\text{act}}, \qquad V = SW^V_{\text{act}}, \tag{4}$$

$$H = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V, \qquad H = (h_1, h_2, \ldots, h_T), \tag{5}$$

$$p_t = \sigma\left(h_t^\top \mathbf{w}_{\text{act}} + b_{\text{act}}\right), \qquad a_t \sim \text{Bernoulli}(p_t), \tag{6}$$

where h_t is the t-th row of H and denotes the attention-enhanced representation of the t-th token. The sigmoid function σ converts the output of the linear layer into a boundary probability $p_t \in (0, 1)$, while $\pi(a_t \mid S; \Theta)$ denotes the probability of selecting action a_t given the complete state sequence S. Moreover, W^Q_{act}, W^K_{act}, and W^V_{act} are learnable projection matrices used to compute the query, key, and value matrices, respectively. The vector $\mathbf{w}_{\text{act}}$ and scalar b_{act} are the weight and bias of the output layer. The actor network policy is applied independently to the source and target branches to generate action sequences a_X and a_Y. Following Struct-XLM [6], a position with $a_j = 1$ marks the end of a new latent phrase. Each single-token phrase is assigned the submatrix [6], and each multi-token phrase is assigned $\mathbf{1} - \mathbf{I}$. The submatrices are assembled into block-diagonal action matrices A_X and A_Y for the source and target branches, respectively. These matrices encode the predicted phrase boundaries and modulate attention in the final layer of the structural encoder.

3.4. Reward Acquirer

The reward acquirer computes rewards as training signals from the source and target structural representations h_X and h_Y produced by the encoder. Following Struct-XLM [6], a translation ranking objective with in-batch negative sampling is adopted. Consider a batch of N parallel sentence pairs $\{(X_i, Y_i)\}_{i=1}^N$, where N denotes the batch size. The token-level structural representations h_i^X and h_j^Y are converted into sentence-level embeddings through mean pooling:

$$z_i^X = \text{Mean}(h_i^X), \qquad z_j^Y = \text{Mean}(h_j^Y). \tag{7}$$

Their similarity is defined as $\phi(h_i^X, h_j^Y) = z_i^X (z_j^Y)^\top$. Furthermore, $I_{ij} \in \{0, 1\}$ is a binary indicator, where $I_{ij} = 1$ indicates that X_i and Y_j form a translation pair, whereas $I_{ij} = 0$ indicates a non-translation pair. The additive margin applied to positive translation pairs is denoted by m. Based on these definitions, the translation ranking loss is computed as follows:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{\sum\limits_{\substack{j=1 \\ I_{ij}=1}}^{N} \exp\big(\phi(h_i^X, h_j^Y) - m\big)}{\sum\limits_{\substack{j=1 \\ I_{ij}=1}}^{N} \exp\big(\phi(h_i^X, h_j^Y) - m\big) + \sum\limits_{\substack{j=1 \\ I_{ij}=0}}^{N} \exp\big(\phi(h_i^X, h_j^Y)\big)}. \tag{8}$$

The batch-level ranking loss does not provide the fine-grained reward signals required for reinforcement learning; therefore, a sentence-level similarity score is derived from this formulation as a reward proxy. For a source representation h_X, the source-to-target score score_X compares h_X with all target representations $\{h_j^Y\}$ in the batch. The indicator I_j specifies whether the corresponding target sentence is a translation of the source sentence. Using the similarity function ϕ, translation indicator I_j, and additive margin m, this score is computed as

$$\text{score}_X = \log \frac{\sum\limits_{\substack{j=1 \\ I_j=1}}^{N} \exp\big(\phi(h_X, h_j^Y) - m\big)}{\sum\limits_{\substack{j=1 \\ I_j=1}}^{N} \exp\big(\phi(h_X, h_j^Y) - m\big) + \sum\limits_{\substack{n=1 \\ I_n=0}}^{N} \exp\big(\phi(h_X, h_j^Y)\big)}. \tag{9}$$

Similarly, score_Y denotes the target-to-source ranking score obtained by comparing h_Y with the source representations in the batch. Thus, the ranking formulation produces two directional scores for each parallel sentence pair: score_X for $X \to Y$, and score_Y for $Y \to X$.

A2C requires reward signals that indicate the optimization direction and reflect the magnitude of improvement to support stable value estimation. Similarity-based scores alone are insufficient for this purpose. Therefore, a contrastive reward is introduced to compare representations generated with and without structural actions. For each parallel sentence pair (X, Y), the structural encoder first generates representations conditioned on the actor network's action matrices. Equation (9) yields the corresponding source-to-target and target-to-source scores, $score^X_{\text{actions}}$ and $score^Y_{\text{actions}}$. Representations are also generated without the action matrices, yielding $score^X_{\text{non-actions}}$ and $score^Y_{\text{non-actions}}$. The rewards are defined as the differences between these scores:

$$\mathcal{R}_x = score^X_{actions} - score^X_{non_actions}\,, \tag{10}$$

$$\mathcal{R}_y = score^Y_{actions} - score^Y_{non_actions}\,. \tag{11}$$

This formulation measures the change in alignment scores attributable to structural actions. A higher score with actions yields a positive reward, a lower score yields a negative reward, and equal scores yield zero reward. The sign and magnitude of this difference provide feedback for value estimation and policy optimization.

3.5. Critic Network

The critic network estimates a state-dependent value function $V_\psi(s_t)$ from the state representations produced by the structural encoder. Under the Siamese framework, the source and target state sequences are processed independently using the same critic network parameters. Given $S_X = (s_1^x, s_2^x, \ldots, s_T^x)$ and $S_Y = (s_1^y, s_2^y, \ldots, s_{T'}^y)$, the critic network produces

a scalar value estimate for every token state, denoted by $V_\psi(s_t^x)$ and $V_\psi(s_t^y)$ for the source and target branches, respectively. Let $\psi = \{\mathbf{w}_{\text{cri}}, b_{\text{cri}}\}$ denote the learnable parameters of the critic network output layer, where $\mathbf{w}_{\text{cri}}$ is its weight vector and b_{cri} is its scalar bias. The operator $\text{Dropout}(\cdot)$ applies dropout regularization to the input state. The value estimates are computed as

$$\begin{aligned} V_\psi(s_t^x) &= \mathbf{w}_{\text{cri}}^\top \text{Dropout}(s_t^x) + b_{\text{cri}}, \\ V_\psi(s_t^y) &= \mathbf{w}_{\text{cri}}^\top \text{Dropout}(s_t^y) + b_{\text{cri}}. \end{aligned} \tag{12}$$

These scalar outputs estimate the expected rewards associated with the token states. During training, they serve as baselines for estimating advantage in the source and target branches, thereby helping to reduce policy gradient variance.

The critic network shares the masked self-attention layer and the subsequent Add&Norm layer with the actor network. Sharing these intermediate representations limits the additional computational overhead of the critic network. The networks nevertheless use different objectives: the actor network learns the policy, while the critic network estimates state values.

3.6. Training Objective

The model is trained under the actor–critic framework, where policy updates are guided by advantage estimation.

3.6.1. Advantage Function

To stabilize training, The TD error [28–30] is adopted as an advantage estimate. Token-level advantages are computed from the delayed rewards $\mathcal{R}_X$ and $\mathcal{R}_Y$ together with the critic network's state value estimates $V_\psi(s_t^x)$ and $V_\psi(s_t^y)$.

Since $\mathcal{R}_X$ and $\mathcal{R}_Y$ are defined at the sentence level, they are projected onto token-level actions. Given action matrices A^X and A^Y, A_t^X and A_t^Y denote the action vectors corresponding to the t-th token in the source and target branches, respectively. For the t-th token, the rewards are defined as follows:

$$r_t^x = R_X \cdot A_t^X, \quad r_t^y = R_Y \cdot A_t^Y. \tag{13}$$

The token-level advantage is the difference between the assigned reward and the state value predicted by the critic network:

$$\hat{A}_t^x = r_t^x - V_\psi(s_t^x), \quad \hat{A}_t^y = r_t^y - V_\psi(s_t^y). \tag{14}$$

3.6.2. Actor Network Loss

The actor network is updated to maximize the expected advantage by increasing the probability of beneficial actions. The loss is computed separately for each Siamese branch, then averaged. For a sequence of T tokens, $t \in \{1, \ldots, T\}$ indexes the token position and M_t denotes the validity mask. Let $\pi_\Theta(a_t \mid S)$ denote the probability of selecting action a_t given the full state sequence S, and let $\widehat{A}_t$ denote the corresponding advantage; then, the loss for each branch is defined as follows:

$$\mathcal{L}_{Actor}(\Theta) = -\frac{1}{\sum_t M_t} \sum_{t=1}^{T} M_t \cdot \log \pi_\Theta(a_t|S) \cdot \hat{A}_t. \tag{15}$$

Let $\mathcal{L}_{Actor}^X$ and $\mathcal{L}_{Actor}^Y$ denote the actor network losses for the source and target branches, respectively. The final actor network loss is their average:

$$\mathcal{L}_{Actor}^{final} = \frac{\mathcal{L}_{Actor}^X + \mathcal{L}_{Actor}^Y}{2}. \tag{16}$$

3.6.3. Critic Network Loss

The critic network minimizes the squared error between its predicted state values and the rewards assigned to token-level actions. For a sequence of T tokens, $t \in \{1, \ldots, T\}$ denotes the token position, $M_t \in \{0, 1\}$ the validity mask, r_t the assigned reward, and $V_\psi(s_t)$ the predicted state value. The critic network loss for each branch is defined as

$$\mathcal{L}_{\text{Critic}}(\psi) = \frac{1}{\sum_{t=1}^{T} M_t} \sum_{t=1}^{T} M_t \Big(V_\psi(s_t) - r_t \Big)^2. \tag{17}$$

Let $\mathcal{L}^X_{\text{Critic}}$ and $\mathcal{L}^Y_{\text{Critic}}$ denote the critic network losses for the source and target branches, respectively. The final critic network loss is obtained by averaging the two branch-specific losses:

$$\mathcal{L}^{\text{final}}_{\text{Critic}} = \frac{\mathcal{L}^X_{\text{Critic}} + \mathcal{L}^Y_{\text{Critic}}}{2}. \tag{18}$$

4. Experiments and Results

4.1. Experimental Details

4.1.1. Baseline Models

1. **InfoXLM** maximizes mutual information to improve semantic alignment and cross-lingual transfer without using RL. The publicly available InfoXLM-large checkpoint is used, with an independently implemented task-specific fine-tuning and evaluation pipeline for the seven XTREME tasks. Since the complete implementation underlying the re-implemented InfoXLM results in the original Struct-XLM study is not publicly available, the results presented in this study should be interpreted as an evaluation under the present implementation as opposed to an exact reproduction of the previously reported results. The comparison between SAC-XLM and InfoXLM evaluates the impact of RL on multilingual models.
2. **mmBERT** is an encoder-only multilingual model trained on large-scale multilingual text without using RL algorithms. Since both mmBERT and SAC-XLM are encoder-only multilingual models, the comparison between them evaluates whether RL-based structural learning can remain competitive with recent advances in large-scale multilingual pretraining.
3. **Struct-XLM** uses RL to discover latent syntactic structures that improve cross-lingual representation alignment. Both Struct-XLM and SAC-XLM use XLM-R$_{large}$ as the structural encoder and are trained within RL frameworks, allowing for a direct comparison of their optimization strategies.

4.1.2. Training Data

The same training corpus as Struct-XLM are used to support a controlled comparison. The corpus contains 13.7K parallel sentence pairs covering 20 languages, with English as the primary source language. It includes parallel Universal Dependencies (UD) treebanks from PUD, ATIS, and ParTUT in UD 2.9 [31], together with the LinES corpus [32]. Table 1 reports the number of sentences in each language.

The relatively small training corpus is adopted for both methodological and practical reasons. First, the proposed structural learning process requires parallel sentences with English as the source language and dependency tree annotations. Publicly available and established resources that jointly satisfy these requirements are limited, particularly for some of the languages considered in this study. Second, SAC-XLM follows the same training corpus as Struct-XLM in order to maintain a controlled comparison between the two training frameworks and minimize the potential influence of differences in training data. Accordingly, the current experiments are intended to examine the optimization behavior of SAC-XLM

under limited parallel supervision rather than to establish generalization to large-scale multilingual training. Its applicability to larger and more diverse dependency-annotated parallel corpora remains to be investigated in future work.

Table 1. Training data statistics by language.

Lang.	Train	Dev	Test
English (en)	12,038	882	800
Swedish (sv)	1634	406	348
Italian (it)	1528	76	109
French (fr)	951	45	83
Turkish (tr)	686	55	40
German (de)	502	20	20
Chinese (zh)	502	20	20
Japanese (ja)	502	20	20
Finnish (fi)	502	20	20
Arabic (ar)	502	20	20
Spanish (es)	502	20	20
Indonesian (id)	502	20	20
Korean (ko)	502	20	20
Thai (th)	502	20	20
Czech (cs)	502	20	20
Russian (ru)	502	20	20
Hindi (hi)	502	20	20
Polish (pl)	502	20	20
Portuguese (pt)	502	20	20
Icelandic (is)	211	20	20

4.1.3. Training Stages

Training follows the three-stage procedure used in Struct-XLM.

1. Stage I: The structural encoder is trained independently for five epochs, with a learning rate of 6×10^{-6}. A warm-up strategy is used to stabilize initial training.
2. Stage II: The structural encoder is frozen, and the actor network and critic network are trained using A2C for two epochs, with a learning rate of 3×10^{-6}.
3. Stage III: All model components are jointly optimized using A2C for five additional epochs, with a learning rate of 3×10^{-6}.

4.1.4. Implementation Details

All experiments are implemented in PyTorch 2.3.1 and run on a single NVIDIA A6000 GPU. The structural encoder is initialized from XLM-R_{large}, and training uses AdamW with a batch size of 5. Learning rates and schedules follow the three-stage training procedure. A dynamic epoch-level early stopping strategy based on validation loss is applied at each training stage. Unless otherwise specified, the training settings follow those used for Struct-XLM to ensure a fair comparison. Table 2 compares the settings reported in the original Struct-XLM study with those used for Struct-XLM and SAC-XLM in this work.

Table 2. Comparison of the implementation and evaluation settings reported in the original Struct-XLM study and those used for Struct-XLM and SAC-XLM in this work.

Setting	Original Struct-XLM	Struct-XLM in This Work	SAC-XLM in This Work
Pretraining settings			
Backbone	XLM-R_{large}	XLM-R_{large}	XLM-R_{large}
Training data	13.7K parallel sentence pairs	13.7K parallel sentence pairs	13.7K parallel sentence pairs
Training stages	Three stages: 5/2/5 epochs	Three stages: 5/2/5 epochs	Three stages: up to 5/2/5 epochs
Learning rates	6×10^{-6}, 3×10^{-6}, and 3×10^{-6}	6×10^{-6}, 3×10^{-6}, and 3×10^{-6}	6×10^{-6}, 3×10^{-6}, and 3×10^{-6}
Batch size	5	5	5
Pretraining early stopping	Not reported	Not applied	Validation loss-based; evaluated every 50 steps; patience = 12
Downstream fine-tuning settings			
Downstream early stopping	Not reported in the paper; patience = 10 in the released code	Patience = 10	Patience = 10
Checkpoint selection	Best development set checkpoint	Best development set checkpoint	Best development set checkpoint
Experimental environment			
GPU	NVIDIA RTX A6000	NVIDIA A6000	NVIDIA A6000

4.2. Evaluation

The models are evaluated on seven XTREME tasks spanning three categories. Sentence classification is evaluated using cross-lingual natural language inference (XNLI) [33] and PAWS-X [34]. Structured prediction is evaluated using part-of-speech (POS) tagging [35] and named entity recognition (NER) [36]. Question answering is evaluated using MLQA [8], XQuAD [37], and the gold-passage version of TyDiQA (TyDiQA-GoldP) [38]. All evaluations use a zero-shot cross-lingual setting in which models are fine-tuned on English data and evaluated across multiple target languages. These tasks assess performance at different linguistic levels, including syntax, semantics, and reasoning.

4.3. Experimental Results

4.3.1. Main Results

Table 3 compares InfoXLM, mmBERT, Struct-XLM, and SAC-XLM on the XTREME benchmark. It reports downstream performance on seven tasks together with the available pretraining and task-specific fine-tuning and evaluation times.

Compared with Struct-XLM, SAC-XLM substantially improves training efficiency. It requires 13.2 h of pretraining, approximately 40% of the 34.4 h required by Struct-XLM. The average per-task fine-tuning time decreases from 27.4 ks to 18.8 ks, a reduction of approximately 26%. These results demonstrate improved efficiency during both pretraining and downstream fine-tuning.

Despite the shorter pretraining time, SAC-XLM preserves overall performance across the seven XTREME tasks. Its reported average score is 69.7, compared with 69.5 for Struct-XLM, 65.1 for InfoXLM, and 64.9 for mmBERT. The InfoXLM results are obtained from the public InfoXLM-large checkpoint using our independently implemented downstream pipeline, and should be interpreted within this evaluation setting; they do not constitute an exact reproduction of the InfoXLM results reported in the original Struct-XLM study. At the task-average level, SAC-XLM outperforms InfoXLM and mmBERT on all seven tasks, suggesting that its efficiency gains are compatible with effective multilingual representation learning.

Table 3. Performance and training time comparison among InfoXLM, mmBERT, Struct-XLM, and SAC-XLM on the XTREME benchmark. The InfoXLM results are obtained using the publicly available InfoXLM-large checkpoint and our independently implemented downstream evaluation pipeline. Time values are reported in thousands of seconds (ks). Bold indicates the best performance (highest scores) and the lowest training time.

Model	XNLI		PAWS-X		POS		NER	
	Acc.	Time	Acc.	Time	F1	Time	F1	Time
InfoXLM	75.4	–	74.6	–	74.0	–	65.4	–
mmBERT	71.7	–	83.7	–	71.3	–	62.8	–
Struct-XLM	78.6	67.1 ks	**88.4**	27.2 ks	75.6	7.7 ks	**68.5**	11.2 ks
SAC-XLM (ours)	**79.6**	**35.9 ks**	88.2	**16.7 ks**	75.6	**5.2 ks**	68.2	**6.7 ks**
	XQuAD		**MLQA**		**TyDiQA**		**Average**	
	F1/EM	Time	F1/EM	Time	F1/EM	Time	P	Time
InfoXLM	72.6/56.6	–	66.2/47.8	–	68.7/50.0	–	65.1	–
mmBERT	73.8/56.5	–	65.4/45.7	–	69.2/49.4	–	64.9	–
Struct-XLM	**78.0**/61.4	**32.7 ks**	**71.5**/52.8	40.3 ks	69.7/50.9	5.8 ks	69.5	27.4 ks
SAC-XLM (ours)	77.8/**62.2**	34.5 ks	71.3/**53.1**	**30.0 ks**	**70.1**/**51.0**	**2.8 ks**	**69.7**	**18.8 ks**
Pretraining time: Struct-XLM: 34.4 h, SAC-XLM (ours): **13.2 h**								

On XNLI, SAC-XLM improves accuracy by 1.0 percentage point over Struct-XLM while using approximately 53% of its fine-tuning time. On TyDiQA, F1 and exact match (EM) increase by 0.4 and 0.1 percentage points, respectively, while the time decreases to approximately 48% of the baseline. SAC-XLM matches Struct-XLM's POS F1 score and reduces the time from 7.7 ks to 5.2 ks. On NER, F1 decreases by 0.3 percentage points, while the time decreases from 11.2 ks to 6.7 ks. Task-specific differences are observed on PAWS-X, XQuAD, and MLQA. On PAWS-X, SAC-XLM's accuracy is 0.2 percentage points lower than that of Struct-XLM, while its time decreases from 27.2 ks to 16.7 ks. On XQuAD, EM increases by 0.8 percentage points and F1 decreases by 0.2 percentage points, with a slightly longer fine-tuning and evaluation time. On MLQA, EM increases by 0.3 percentage points, F1 decreases by 0.2 percentage points, and time decreases from 40.3 ks to 30.0 ks. The differing changes in F1 and EM may reflect variations in predicted answer boundaries rather than a consistent decline in answering accuracy. Although performance decreases slightly on some tasks, these variations do not result in an overall degradation in downstream performance.

4.3.2. Detailed Results for Downstream Tasks

Tables 4–10 report the per-language results for InfoXLM, mmBERT, Struct-XLM, and SAC-XLM. Sentence classification results are presented for XNLI and PAWS-X in Tables 4 and 5; structured prediction results for POS and NER appear in Tables 6 and 7; and question-answering results for XQuAD, MLQA, and TyDiQA appear in Tables 8, 9, and 10, respectively.

Table 4. XNLI accuracy by language. Bold indicates the best performance (highest scores).

Lang.	en	ar	bg	de	el	es	fr	hi	ru	sw	th	tr	ur	vi	zh	AVG
InfoXLM	86.1	72.9	78.2	77.9	76.6	80.2	79.5	71.5	77.1	66.4	72.8	73.8	66.2	75.9	76.0	75.4
mmBERT	83.6	68.6	74.2	76.4	71.3	79.0	78.1	64.5	72.9	62.5	66.9	70.7	60.3	74.0	72.5	71.7
Struct-XLM	**88.1**	77.4	82.1	81.9	81.1	83.2	81.4	74.1	78.1	70.5	76.3	77.6	71.3	79.2	78.1	78.6
SAC-XLM	88.0	**78.8**	**82.5**	**82.6**	**81.7**	**83.8**	**83.0**	**76.5**	**79.5**	**71.5**	**76.6**	**78.4**	**72.6**	**79.4**	**79.0**	**79.6**

Table 5. PAWS-X accuracy by language. Bold indicates the best performance (highest scores).

Lang.	en	de	es	fr	ja	ko	zh	AVG
InfoXLM	82.1	76.4	76.5	78.8	67.5	70.6	70.6	74.6
mmBERT	92.9	86.4	88.9	89.7	78.2	71.0	79.2	83.7
Struct-XLM	**96.0**	**92.0**	91.3	91.6	**82.0**	81.7	**84.4**	**88.4**
SAC-XLM	95.7	90.8	**92.4**	**92.2**	80.6	**82.1**	83.9	88.2

Table 6. POS F1 scores by language. Bold indicates the best performance (highest scores).

Lang.	en	af	ar	bg	de	el	es	et	eu	fa	fi	fr	he	hi	hu	id	it
InfoXLM	95.9	87.8	64.3	88.6	**89.4**	**88.4**	88.5	84.7	71.3	70.4	83.9	87.3	63.3	66.3	83.1	72.3	88.6
mmBERT	94.0	88.6	59.3	83.4	87.4	87.3	79.9	83.6	67.0	70.3	77.7	81.0	63.9	66.2	79.7	**73.9**	82.0
Struct-XLM	**96.0**	88.4	66.2	88.4	89.1	86.8	89.2	**87.3**	**77.4**	70.3	85.5	**88.2**	64.3	**75.6**	82.4	72.8	89.9
SAC-XLM	95.9	**89.2**	**68.9**	**89.4**	89.0	86.3	**89.5**	87.2	66.5	**72.1**	**86.9**	**88.2**	**70.4**	71.9	**83.3**	72.5	**90.1**
Lang.	ja	kk	ko	mr	nl	pt	ru	ta	te	th	tl	tr	ur	vi	yo	zh	AVG
InfoXLM	35.6	76.2	52.7	81.2	89.2	87.5	89.4	64.9	84.8	48.5	89.3	75.1	60.1	57.5	32.5	43.3	74.0
mmBERT	35.9	73.9	48.1	79.4	88.4	83.2	86.8	62.0	80.7	40.8	88.4	74.4	58.9	57.8	31.6	38.9	71.3
Struct-XLM	**36.5**	**78.3**	51.7	83.2	89.2	88.6	90.0	**64.9**	**86.0**	51.6	**94.1**	**75.9**	**8.4**	**59.3**	**34.2**	**44.9**	**75.6**
SAC-XLM	33.0	77.3	**53.7**	**83.3**	**89.6**	**88.9**	**90.1**	64.6	83.0	**51.7**	91.6	75.2	65.6	57.7	23.0	44.7	**75.6**

Table 7. NER F1 scores by language. Bold indicates the best performance (highest scores).

Lang.	en	af	ar	bg	bn	de	el	es	et	eu	fa	fi	fr	he	hi	hu	id	it	ja	jv
InfoXLM	82.4	76.5	**61.8**	81.3	72.8	79.1	76.0	72.1	68.9	54.2	54.0	75.1	79.6	52.8	67.7	74.0	53.2	79.6	24.7	56.1
mmBERT	81.9	74.2	42.6	82.5	76.8	83.1	71.2	65.0	74.7	50.3	54.0	71.4	76.6	52.1	73.2	71.8	49.0	77.6	18.4	66.3
Struct-XLM	**91.2**	**86.5**	49.1	**88.0**	79.4	**85.9**	82.1	**79.3**	**86.8**	**72.0**	**64.4**	**88.8**	**87.6**	57.1	73.9	**86.4**	58.2	**88.7**	21.8	**70.0**
SAC-XLM	90.9	84.1	49.8	85.5	**84.5**	83.6	**84.1**	76.5	81.7	64.9	60.3	85.3	86.4	**61.6**	**77.2**	83.7	**59.2**	86.9	**27.9**	67.4
Lang.	ka	kk	ko	ml	mr	ms	my	nl	pt	ru	sw	ta	te	th	tl	tr	ur	vi	yo	zh	AVG
InfoXLM	71.2	45.6	54.4	68.1	61.8	72.3	57.2	82.1	79.0	65.3	71.6	58.8	53.4	10.4	75.5	77.1	**71.4**	77.5	53.6	30.2	65.4
mmBERT	65.5	41.7	51.6	59.5	57.8	65.0	55.9	79.4	78.7	58.5	73.2	52.4	49.9	6.1	72.8	74.7	70.5	77.9	59.8	20.8	62.8
Struct-XLM	69.9	**52.5**	59.9	68.8	**74.5**	69.3	57.7	**90.8**	**88.6**	**72.6**	**79.9**	62.7	61.4	**11.0**	**83.1**	84.4	66.3	**81.8**	**65.0**	28.3	**68.5**
SAC-XLM	**71.9**	48.1	**64.3**	**71.3**	66.1	**78.5**	**64.4**	89.3	85.8	69.5	78.5	**66.2**	**61.9**	10.6	79.7	**84.6**	66.0	81.5	48.8	**34.7**	68.2

Table 8. XQuAD results (F1/EM) by language.

Lang.	en	ar	de	el	es	hi	ru	th	tr	vi	zh	AVG
InfoXLM	70.3/52.9	74.2/57.0	73.8/54.4	84.1/72.1	77.2/58.4	69.5/51.7	73.1/56.1	68.0/58.9	69.3/53.5	74.5/54.1	64.7/53.4	72.6/56.6
mmBERT	**79.0**/**65.9**	71.6/51.3	76.7/58.7	72.4/56.2	79.9/59.5	73.0/56.6	73.0/56.2	72.7/58.3	**75.8**/58.4	75.1/55.0	62.7/48.0	73.8/56.5
Struct-XLM	74.2/56.1	**79.7**/**62.3**	79.1/60.5	**87.2**/**75.9**	**81.2**/**61.7**	**75.4**/**57.9**	**79.6** /61.8	74.4/60.8	**75.8**/**58.7**	**79.4**/**60.0**	**71.8**/59.7	**78.0**/61.4
SAC-XLM	74.8/58.2	78.9/62.0	**80.3**/**62.3**	86.4/75.2	80.6/61.6	75.1/**57.9**	79.1/**62.9**	**75.5**/**65.5**	74.7/58.2	78.7/58.8	71.2/**62.0**	77.8/**62.2**

Table 9. MLQA results (F1/EM) by language. Bold indicates the best performance (highest scores).

Lang.	en	ar	de	es	hi	vi	zh	AVG
InfoXLM	58.6/39.4	63.9/48.8	79.6/65.8	68.3/49.5	63.9/45.7	67.4/46.7	61.9/38.7	66.2/47.8
mmBERT	**74.6**/**59.5**	64.4/42.5	76.7/59.4	70.7/51.6	53.2/34.8	66.0/45.3	52.1/27.1	65.4/45.7
Struct-XLM	65.4/44.5	**69.3**/**53.6**	**83.4**/69.7	**73.2**/54.7	69.6/52.1	72.8/51.5	**66.8**/**43.2**	**71.5**/52.8
SAC-XLM	64.9/**44.6**	68.3/53.5	83.2/**70.0**	72.8/**54.8**	**69.7**/**52.2**	**73.2**/**52.2**	66.7/44.1	71.3/**53.1**

Table 10. TyDiQA-GoldP results (F1/EM) by language. Bold indicates the best performance (highest scores).

Lang.	en	ar	bn	fi	id	ko	ru	sw	te	AVG
InfoXLM	72.7/59.5	75.2/52.4	59.5/40.7	72.9/55.6	79.0/63.0	55.7/43.8	64.5/35.5	67.2/47.5	71.2/52.2	68.7/50.0
mmBERT	71.0/55.9	**74.5**/49.3	65.3/45.3	72.4/50.8	72.7/50.8	60.3/45.9	64.3/**39.1**	69.6/**50.6**	72.5/**56.7**	69.2/49.4
Struct-XLM	**73.8**/**62.0**	72.1/49.2	**66.1**/**49.6**	68.5/50.9	77.9/59.1	**61.2**/**47.8**	65.4/35.5	**69.8**/49.5	**72.9**/54.1	69.7/50.9
SAC-XLM	72.8/60.0	74.2/**53.6**	64.7/44.2	**73.4**/**56.1**	**81.6**/**65.1**	57.2/40.9	**66.0**/37.2	68.6/49.9	72.0/51.6	**70.1**/**51.0**

4.4. Stability Analysis

4.4.1. Loss Curves During Pretraining

The pretraining loss curves of SAC-XLM and Struct-XLM are compared to assess training stability. Both models execute 12,035 steps in stage I and 4814 steps in stage II. The corresponding times are 5808.22 s and 32,202.37 s for Struct-XLM, and 4219.97 s and 23,236.76 s for SAC-XLM. As the differences in these stages are relatively limited, Figure 3 focuses on stage III. In this stage, Struct-XLM executes 12,035 steps in 85,860.55 s, with an average cost of approximately 7.13 s per step. SAC-XLM triggers early stopping at step 2500 and requires 20,172.92 s, with an average cost of 8.06 s per step. Therefore, its reduction in training time comes from executing fewer steps, despite a slightly higher per-step cost due to additional reward computation. SAC-XLM executes 79.2% fewer steps in stage III, reducing training time for this stage by 76.5%. Figure 3 also shows a clear difference in convergence behavior. Struct-XLM exhibits noticeable fluctuations throughout training, including irregular loss spikes and a slower overall convergence trend. Over approximately 12,000 steps, its loss continues to oscillate around 2.28, indicating unstable optimization. Its initial and final losses are 2.29 and 2.26, respectively, both higher than those of SAC-XLM.

SAC-XLM exhibits a smoother and more stable convergence pattern. Its loss decreases steadily with fewer oscillations and stabilizes below 2.21 by approximately step 2000, while Struct-XLM continues to fluctuate. SAC-XLM satisfies the early stopping criterion after approximately 2500 steps, reaching a stable optimization state with substantially fewer updates. This behavior can be attributed to its A2C-based optimization mechanism, which uses value-based feedback to reduce noisy updates and produce smoother training dynamics.

The loss curves suggest that SAC-XLM requires fewer training steps and exhibits smaller fluctuations during pretraining. The next subsection examines relative variability using the coefficient of variation.



Figure 3. Training loss by optimization step for Struct-XLM and SAC-XLM during pretraining stage III. SAC-XLM converges more smoothly and quickly, while Struct-XLM exhibits pronounced oscillations throughout training.

4.4.2. Coefficient of Variation

The coefficient of variation (CV) [39] is used to quantify the relative variability of model updates during stage III. Defined as the ratio of the standard deviation to the mean, the CV provides a normalized measure of dispersion. Lower values indicate more consistent and stable updates, while higher values indicate greater variability. The actor network in SAC-XLM and the policy network in Struct-XLM fulfill the same role in determining and updating the policy, making their CV values a direct basis for comparison.

Figure 4 shows that Struct-XLM's policy network exhibits high and fluctuating CV values throughout training. Values frequently exceed 1, with large spikes and irregular oscillations across most of the training trajectory. This variability indicates unstable parameter updates and inconsistent optimization behavior. By contrast, the CV values for SAC-XLM's actor network remain below 0.05 throughout training (Figure 5). The nearly flat curve shows only minor fluctuations, indicating stable and controlled parameter updates.

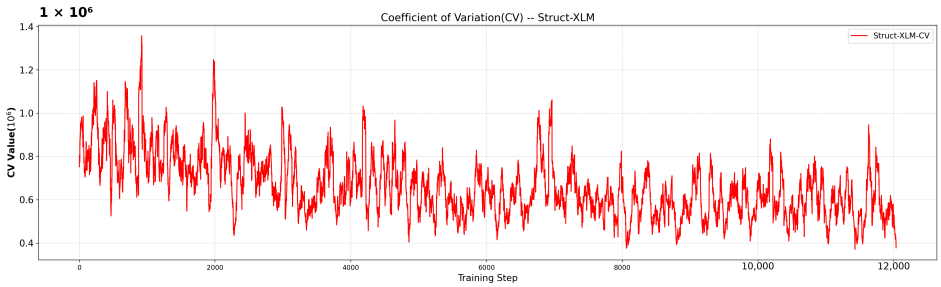

Figure 4. CV of the policy network in Struct-XLM during pretraining stage III.

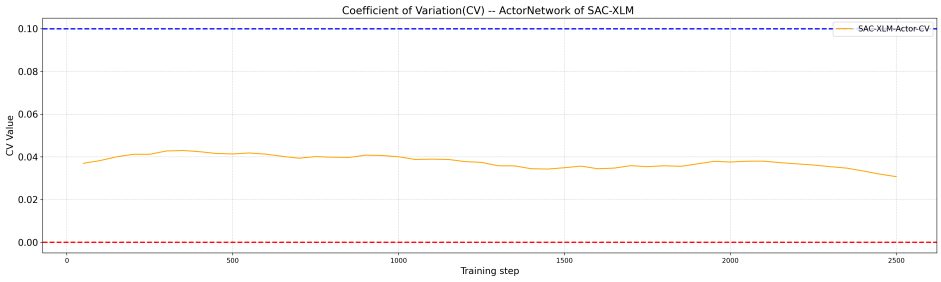

Figure 5. CV of the actor network parameters in SAC-XLM during pretraining stage III. Values remain below 0.05, indicating stable policy updates.

The difference between the CV trajectories suggests that SAC-XLM maintains a more stable optimization process, consistent with its smoother loss curve. These observations also align with the design of A2C, which uses value-guided updates to reduce instability from noisy gradient signals and produce more consistent training dynamics.

5. Theoretical Insights into Stability and Efficiency

While the empirical improvements of SAC-XLM arise from a combination of factors, including architectural factors, reward formulation, and the optimization algorithm. This section focuses specifically on the theoretical role of the actor–critic optimization framework in reducing gradient variance. The following analysis is intended to provide a theoretical intuition for the observed optimization stability of SAC-XLM. The final performance and stability of SAC-XLM are influenced by the combined effects of the optimization algorithm, reward formulation, model framework, and training strategy.

5.1. Gradient Variability Analysis

The variability of the policy gradient estimators used in Struct-XLM and SAC-XLM are examined. For Struct-XLM, the estimator is written as follows:

$$g_{Struct-XLM} = \nabla \mathcal{J}(\Theta) = \sum_{t=1}^{T} R_L \nabla_\Theta \log \pi_\Theta(a_t|s_t), \tag{19}$$

while for SAC-XLM the gradient is expressed as

$$g_{SAC-XLM} = \nabla \mathcal{J}(\Theta) = \sum_{t=1}^{T} (R_L - V(s_t)) \nabla_\Theta \log \pi_\Theta(a_t|s_t), \tag{20}$$

where R_t denotes the reward at step t and $\pi_\Theta(a_t|s_t)$ is the probability of selecting action a_t in state s_t. The symbols a_t and s_t denote the action and state at step t, respectively. The state value estimate $V(s_t)$ serves as the baseline $b(t)$.

CV is used to compare the relative variability of the two stochastic gradient estimators. Lower CV values generally correspond to more stable gradient updates and more consistent optimization behavior. CV is defined as follows:

$$CV = \frac{\sigma}{\mu} = \frac{\sqrt{\mathrm{Var}(g)}}{\mathbb{E}[|g|]} = \frac{\sqrt{\mathbb{E}[g^2] - (\mathbb{E}[g])^2}}{\mathbb{E}[|g|]} \tag{21}$$

where $\sigma = \sqrt{\mathrm{Var}(g)}$ denotes the standard deviation of the gradient estimator and $\mu = \mathbb{E}[|g|]$ represents its expected magnitude.

Struct-XLM uses REINFORCE, while SAC-XLM uses A2C. Both are policy gradient methods that directly optimize expected rewards using terms of the form $\nabla_\Theta \log \pi_\Theta(a_t \mid s_t)$. Their gradient estimators are unbiased, that is,

$$\mathbb{E}[g] = \nabla J(\Theta). \tag{22}$$

This property implies that both methods optimize the same expected objective, with differences in CV primarily associated with the variance of their gradient estimators. The baseline $b(t)$ depends only on the state s_t and is independent of the action a_t. According to the policy gradient theorem [40], introducing such a baseline does not change the expected gradient because the baseline term has zero expected contribution; therefore,

$$\mathbb{E}[g_{\text{Struct-XLM}}] = \mathbb{E}[g_{\text{SAC-XLM}}] = \nabla J(\Theta). \tag{23}$$

An important distinction between the two optimization algorithms is the variance of their gradient estimators. Under this theoretical formulation, the ratio of their CV values is expressed as follows:

$$\frac{CV_{\text{Struct-XLM}}}{CV_{\text{SAC-XLM}}} = \sqrt{\frac{\mathrm{Var}(g_{\text{Struct-XLM}})}{\mathrm{Var}(g_{\text{SAC-XLM}})}}. \tag{24}$$

Actor–critic methods introduce a state-dependent baseline in order to reduce policy-gradient variance without changing the expected gradient. The following derivation illustrates this variance reduction property by comparing the estimators used in Struct-XLM and SAC-XLM:

$$\begin{aligned}\Delta &= \mathrm{Var}(g_{\text{Struct-XLM}}) - \mathrm{Var}(g_{\text{SAC-XLM}}) \\ &= 2\mathbb{E}\left[\left(\sum_{t=1}^{T} R_t l_t\right)\left(\sum_{t=1}^{T} b(t) l_t\right)\right] - \mathbb{E}\left[\left(\sum_{t=1}^{T} b(t) l_t\right)^2\right] \geq 0. \end{aligned} \tag{25}$$

The baseline $b(t) = V(s_t)$ approximates the expected return; subtracting it centers the return estimator around its expectation, thereby reducing the variability of the policy gradient estimator while preserving its expected value. This yields the following relationship:

$$\mathrm{Var}(g_{\text{Struct-XLM}}) \geq \mathrm{Var}(g_{\text{SAC-XLM}}). \tag{26}$$

The total gradient variance of Struct-XLM and SAC-XLM during stage III is also estimated. The variance term in Equation (21) is evaluated for the estimators in Equations (19) and (20). Figure 6 shows the resulting trajectories under the experimental settings used here.



Figure 6. Estimated total gradient variance for Struct-XLM and SAC-XLM during pretraining stage III.

Figure 6 shows different gradient variance trajectories for the two models. Within the overlapping range of recorded steps, SAC-XLM's variance is concentrated in a lower and narrower range, while Struct-XLM starts at a higher level and decreases with more noticeable fluctuations. This pattern provides complementary empirical context for the variance reduction intuition in Equations (25) and (26), although it should be interpreted as an observation under the current experimental settings rather than as verification of the theoretical relationship or an isolated assessment of the actor–critic mechanism. The analysis suggests that A2C's variance reduction property leads to lower gradient variance, and consequently to a smaller CV:

$$CV_{\text{Struct-XLM}} > CV_{\text{SAC-XLM}}. \tag{27}$$

The theoretical discussion and observed variance trajectories suggest that the state-dependent baseline may contribute to SAC-XLM's lower gradient variability. Nevertheless, the present analysis is intended only to offer a possible interpretation of this behavior rather than to establish a direct causal relationship; there are also other differences between SAC-XLM and Struct-XLM, including the reward formulation, network architecture, and training procedure.

5.2. Gradient Variance and Optimization Efficiency

In stochastic optimization, lower gradient variance is generally associated with more stable updates and fewer steps to convergence. To illustrate this relationship, consider the standard stochastic update

$$\Theta_{k+1} = \Theta_k + \eta_k g_k. \tag{28}$$

This update is a stochastic approximation process, since the gradient estimator is random. Its convergence behavior can be analyzed using stochastic optimization theory [41]. Existing analyses bound the expected squared gradient norm to characterize progress toward a stationary point. This bound depends on gradient variance, and can be written as follows:

$$\mathbb{E}\left[\|\nabla J(\Theta)\|^2\right] \leq O\left(\frac{L}{\sum_{k=0}^{T-1}\eta_k}\right) + O\left(\frac{\sigma^2\sum_{k=0}^{T-1}\eta_k^2}{\sum_{k=0}^{T-1}\eta_k}\right) \tag{29}$$

where L is a constant related to the initial function value, η_k is the learning rate, and $\sigma^2 = \mathrm{Var}(g)$ represents the gradient variance.

To characterize the number of steps needed to reach a target accuracy, consider $\mathbb{E}[\|\nabla J(\Theta)\|^2] \leq \epsilon^2$. The learning rate is typically chosen as $\eta_k \propto 1/\sqrt{T}$ in order to balance the bias and variance terms in the convergence bound. Substituting this schedule into Equation (29) yields the following relationship:

$$\epsilon \propto \frac{\sigma}{\sqrt{T}} \Rightarrow T \propto \frac{\sigma^2}{\epsilon^2}. \tag{30}$$

Under the assumptions of stochastic optimization theory, this relationship suggests that reducing gradient variance decreases the number of steps needed to achieve the same optimization accuracy. A2C's variance reduction property offers one plausible explanation for the improved efficiency observed in SAC-XLM. Together with the preceding analysis, this discussion provides theoretical intuition for the observed optimization behavior rather than a complete proof of the empirical improvements.

6. Analysis

6.1. Random Seed Analysis

The repeated downstream experiments on all seven XTREME tasks use three random seeds: 42, 123, and 2024. Table 11 reports the individual results and their means and standard deviations. Variation across runs differs by task and metric. Among the ten reported metrics, SAC-XLM has a lower standard deviation than Struct-XLM on seven, the same reported standard deviation on XQuAD EM and MLQA F1, and a slightly higher standard deviation on TyDiQA F1. For XNLI, SAC-XLM achieves 79.4 ± 0.7, compared with 78.4 ± 1.0 for Struct-XLM. Its standard deviations on PAWS-X and NER are also lower. These results suggest lower sensitivity to random initialization on several metrics, although the pattern is not uniform. With only three seeds, this analysis provides only preliminary evidence of this behavior.

Table 11. Performance of Struct -XLM and SAC-XLM on seven XTREME tasks under different random seeds. Results are reported as accuracy for XNLI and PAWS-X, F1 for POS and NER, and F1/EM for the question-answering tasks. Bold indicates the best performance (highest scores).

Seed	Model	XNLI Acc.	PAWS-X Acc.	POS F1	NER F1	XQuAD F1/EM	MLQA F1/EM	TyDiQA F1/EM
42	Struct-XLM	78.6	**88.4**	**75.6**	**68.5**	**78.0**/61.4	**71.5**/52.8	69.7/50.9
	SAC-XLM	**79.6**	88.2	**75.6**	68.2	77.8/**62.2**	71.3/**53.1**	**70.1**/**51.0**
123	Struct-XLM	77.4	87.7	**76.6**	52.2	76.0/60.6	**69.2**/49.6	68.8/48.7
	SAC-XLM	**80.0**	**87.8**	75.5	**54.7**	**77.0**/**61.4**	69.0/**50.2**	**69.3**/**50.8**
2024	Struct-XLM	**79.3**	87.2	**75.7**	53.2	**77.1**/60.9	**70.8**/50.7	**68.8**/49.3
	SAC-XLM	78.6	**88.0**	**75.7**	**66.3**	76.8/**61.5**	69.8/**52.5**	68.7/**52.2**
Mean ± Std.	Struct-XLM	78.4 ± 1.0	87.8 ± 0.6	**76.0 ± 0.6**	58.0 ± 9.14	**77.0 ± 1.0**/61.0 ± 0.4	**70.5 ± 1.2**/51.0 ± 1.6	69.1 ± 0.5/49.6 ± 1.1
	SAC-XLM	**79.4 ± 0.7**	**88.0 ± 0.2**	75.6 ± 0.1	**63.0 ± 7.3**	77.2 ± 0.5/**61.7 ± 0.4**	70.0 ± 1.2/**51.9 ± 1.5**	**69.4 ± 0.7**/**51.3 ± 0.8**

6.2. Ablation Study

Evaluations for four ablation settings are performed to examine the contributions of structural actions, the critic, and the proposed reward. In the first setting, *w/o action*, the learned action matrix is replaced with a random matrix. This removes the learned structural guidance, but may also disrupt attention connections; thus, any performance loss cannot be attributed solely to the absence of learned structure. The second setting, *all-pass action*, replaces the learned matrix throughout training with a fixed all-ones matrix. This preserves unrestricted token interactions while removing learned structural selection. Comparing these two settings helps to assess the effect of random attention corruption, while comparing the all-pass setting with the full model assesses the additional contribution of learned structural actions.

The third setting, *w/o critic*, removes the critic network and optimizes the policy without a learned value baseline. This tests the contribution of value-based policy optimization. The fourth setting, *w/o reward*, replaces the proposed reward with the translation ranking reward used in Struct-XLM in order to test the effect of the reward formulation.

Table 12 shows that all four ablation settings reduce overall performance relative to the full SAC-XLM model. Replacing the learned action matrix with a random matrix decreases the overall average score from 69.7 to 66.8, while an all-pass matrix yields 68.4. These results indicate that neither random connections nor unrestricted token interactions provide the same benefits as the learned structural interactions, supporting the contribution of RL-guided structure discovery. Removing the critic network also degrades performance, indicating that actor–critic optimization is more effective than using the actor network alone. Replacing the proposed reward with the translation ranking reward reduces downstream performance, demonstrating the value of the proposed reward as an optimization signal.

Table 12. Ablation results for SAC-XLM.

Model	XNLI	PAWS-X	POS	NER	XQuAD	MLQA	TyDiQA	Overall
	Acc.	Acc.	F1	F1	F1/EM	F1/EM	F1/EM	Avg.
SAC-XLM	79.6	88.2	75.6	68.2	77.8/62.2	71.3/53.1	70.1/51.0	69.7
w/o action	78.2	65.1	74.4	67.8	77.1/63.0	70.8/51.9	69.1/51.0	66.8
all-pass action	78.8	80.8	75.4	68.1	77.0/62.5	70.2/51.3	69.2/50.9	68.4
w/o critic	78.4	62.3	75.1	68.0	76.4/63.0	70.7/52.6	69.3/51.7	66.7
w/o reward	79.0	78.6	75.0	67.6	76.9/61.4	70.0/52.7	69.4/50.7	68.1

6.3. Backbone Analysis

Additional experiments with mBERT examine the effect of the multilingual backbone. As shown in Table 13, both Struct-XLM and SAC-XLM perform less well with mBERT than with XLM-R_{large}. This is primarily because mBERT has fewer parameters and weaker multilingual representation capabilities, resulting in less expressive cross-lingual structural representations. Nevertheless, SAC-XLM maintains comparable performance to Struct-XLM while requiring less training time. These results suggest that performance depends on backbone capacity, while the proposed actor–critic framework remains effective for optimization with both evaluated backbones.

Table 13. Pretraining time and XNLI accuracy with different multilingual backbones. Bold indicates the best performance (highest scores).

Backbone	Method	Pre-Training Time (h)	XNLI Acc.
mBERT	Struct-XLM	15.0	66.4
	SAC-XLM (ours)	**7.0**	**66.8**
XLM-R_{large}	Struct-XLM	34.4	78.6
	SAC-XLM (ours)	**13.2**	**79.6**

7. Conclusions

This paper proposes SAC-XLM, an A2C-based optimization framework for multilingual structure learning. Its structure discovery mechanism enables more stable and efficient learning of multilingual structural representations. SAC-XLM reduces pretraining time to approximately 40% of that required by Struct-XLM without substantially sacrificing performance. Because training uses the same limited parallel data setting as Struct-XLM, the generalizability of the learned structures still requires further investigation. The framework's effectiveness may also vary with the capacity of the multilingual backbone. Future work will explore larger-scale multilingual training, more powerful multilingual models, and multimodal multilingual applications.

Author Contributions: Conceptualization, Y.H.; methodology, Y.H. and S.-B.P.; validation, Y.H. and G.-M.P.; formal analysis, Y.H.; investigation, Y.H. and S.-B.P.; writing—original draft preparation, Y.H.; writing—review and editing, Y.H., G.-M.P. and S.-B.P.; visualization, Y.H.; supervision, S.-B.P.; project administration, Y.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2022-II0951, Development of Uncertainty-Aware Agents Learning by Asking Questions, 40%) and the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2026-RS-2023-00258649) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation, 40%) and the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00509257, Global AI Frontier Lab, 20%).

Data Availability Statement: The pretraining datasets are available through Universal Dependencies (http://universaldependencies.org/; accessed on 11 November 2021) and the following resources: PUD (http://universaldependencies.org/conll17/; accessed on 27 September 2017), ATIS (https://github.com/howl-anderson/ATIS_dataset/; accessed on 21 November 2018), ParTUT (https://github.com/msang/partut-repo; accessed on 1 January 2015), and LinES (https://aclanthology.org/W07-2441.pdf; accessed on 23 May 2007). The datasets for the seven downstream tasks are publicly available through XTREME (https://sites.research.google/xtreme; accessed on 13 April 2020).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Pires, T.; Schlinger, E.; Garrette, D. How Multilingual is Multilingual BERT? In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 4996–5001. [CrossRef]
2. Koroteev, M.V. BERT: A Review of Applications in Natural Language Processing and Understanding. *arXiv* **2021**, arXiv:2103.11943. [CrossRef]
3. Conneau, A.; Lample, G. Cross-lingual Language Model Pretraining. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Volume 32.

4. Conneau, A.; Khandelwal, K.; Goyal, N.; Chaudhary, V.; Wenzek, G.; Guzmán, F.; Grave, E.; Ott, M.; Zettlemoyer, L.; Stoyanov, V. Unsupervised Cross-lingual Representation Learning at Scale. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 8440–8451. [CrossRef]
5. Kudo, T.; Richardson, J. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv* **2018**, arXiv:1808.06226. [CrossRef]
6. Wu, L.; Lu, W. Struct-XLM: A Structure Discovery Multilingual Language Model for Enhancing Cross-lingual Transfer through Reinforcement Learning. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, 6–10 December 2023; pp. 3405–3419. [CrossRef]
7. Ahmad, W.; Li, H.; Chang, K.W.; Mehdad, Y. 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), Online, 1–6 August 2021; pp. 4538–4554. [CrossRef]
8. Lewis, P.; Oguz, B.; Rinott, R.; Riedel, S.; Schwenk, H. MLQA: Evaluating Cross-lingual Extractive Question Answering. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 7315–7330. [CrossRef]
9. Williams, R.J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **1992**, *8*, 229–256. [CrossRef]
10. Metropolis, N.; Ulam, S. The Monte Carlo method. *J. Am. Stat. Assoc.* **1949**, *44*, 335–341. [CrossRef] [PubMed]
11. Forbes, G.C.; Gupta, N.; Villalobos-Arias, L.; Potts, C.M.; Jhala, A.; Roberts, D.L. Potential-based reward shaping for intrinsic motivation. *arXiv* **2024**, arXiv:2402.07411. [CrossRef]
12. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347. [CrossRef]
13. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016; pp. 1928–1937.
14. Siddhant, A.; Hu, J.; Johnson, M.; Firat, O.; Ruder, S. XTREME: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. In Proceedings of the International Conference on Machine Learning, Virtual, 13–18 July 2020; pp. 4411–4421.
15. Ebrahimi, A.; von der Wense, K. How to adapt your pretrained multilingual model to 1600 languages. 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), Online, 1–6 August 2021; pp. 4555–4567. [CrossRef]
16. Qiu, P.; Wu, C.; Zhang, X.; Lin, W.; Wang, H.; Zhang, Y.; Wang, Y.; Xie, W. Towards building multilingual language model for medicine. *Nat. Commun.* **2024**, *15*, 8384. [CrossRef] [PubMed]
17. Liu, Y.; Gu, J.; Goyal, N.; Li, X.; Edunov, S.; Ghazvininejad, M.; Lewis, M.; Zettlemoyer, L. Multilingual Denoising Pre-training for Neural Machine Translation. *Trans. Assoc. Comput. Linguist.* **2020**, *8*, 726–742. [CrossRef]
18. Xue, L.; Constant, N.; Roberts, A.; Kale, M.; Al-Rfou, R.; Siddhant, A.; Barua, A.; Raffel, C. mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 6–11 June 2021; pp. 483–498. [CrossRef]
19. Zhang, Z.R.; Tan, C.; Huang, S.; Huang, F. Veco 2.0: Cross-lingual language model pre-training with multi-granularity contrastive learning. *arXiv* **2023**, arXiv:2304.08205. [CrossRef]
20. Chi, Z.; Dong, L.; Wei, F.; Yang, N.; Singhal, S.; Wang, W.; Song, X.; Mao, X.L.; Huang, H.; Zhou, M. InfoXLM: An Information-Theoretic Framework for Cross-Lingual Language Model Pre-Training. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 6–11 June 2021; pp. 3576–3588.. [CrossRef]
21. Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H.W.; Sutton, C.; Gehrmann, S.; et al. PaLM: Scaling Language Modeling with Pathways. *J. Mach. Learn. Res.* **2023**, *24*, 1–113.
22. Barham, P.; Chowdhery, A.; Dean, J.; Ghemawat, S.; Hand, S.; Hurt, D.; Isard, M.; Lim, H.; Pang, R.; Roy, S.; et al. Pathways: Asynchronous distributed dataflow for ml. *Proc. Mach. Learn. Syst.* **2022**, *4*, 430–449.
23. Lin, X.V.; Mihaylov, T.; Artetxe, M.; Wang, T.; Chen, S.; Simig, D.; Ott, M.; Goyal, N.; Bhosale, S.; Du, J.; et al. Few-shot Learning with Multilingual Generative Language Models. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Abu Dhabi, United Arab Emirates, 7–11 December 2022; pp. 9019–9052. [CrossRef]
24. Marone, M.; Weller, O.; Fleshman, W.; Yang, E.; Lawrie, D.; Van Durme, B. mmBERT: A modern multilingual encoder with annealed language learning. *arXiv* **2025**, arXiv:2509.06888. [CrossRef]
25. Team, N.; Costa-jussà, M.R.; Cross, J.; Çelebi, O.; Elbayad, M.; Heafield, K.; Heffernan, K.; Kalbassi, E.; Lam, J.; Licht, D.; et al. No language left behind: Scaling human-centered machine translation. *arXiv* **2022**, arXiv:2207.04672. [CrossRef]

26. Luo, X.; Zhang, Y.; Zhang, M.; Xiao, K.; Huang, W.; Li, Z. Noise-augmented multi-modal entity alignment with confidence-based dynamic fusion. *Sci. China Inf. Sci.* **2026**, *69*, 172105. [CrossRef]
27. Tang, J.; Li, Z.; Yang, J.; Lin, D.; Zhang, B.; Yang, G.; Fang, L. GeoTP: Parameter-level intelligent planner of GIS tool-use chains based on instruction tuning of large language models. *Knowl. Eng. Rev.* **2026**, *41*, e012. [CrossRef]
28. Rolls, E.T.; McCabe, C.; Redoute, J. Expected Value, Reward Outcome, and Temporal Difference Error Representations in a Probabilistic Decision Task. *Cereb. Cortex* **2007**, *18*, 652–663. [CrossRef] [PubMed]
29. Sutton, R.S. Learning to predict by the methods of temporal differences. *Mach. Learn.* **1988**, *3*, 9–44. [CrossRef]
30. Starkweather, C.K.; Uchida, N. Dopamine signals as temporal difference errors: Recent advances. *Curr. Opin. Neurobiol.* **2021**, *67*, 95–105. [CrossRef] [PubMed]
31. Zeman, D.; Nivre, J.; Abrams, M.; Ackermann, E.; Aepli, N.; Aghaei, H.; Agić, Ž.; Ahmadi, A.; Ahrenberg, L.; Ajede, C.K.; et al. *Universal Dependencies 2.9*; LINDAT/CLARIAH-CZ Digital Library at the Institute of Formal and Applied Linguistics (ÚFAL); Universal Dependencies Consortium: Prague, Czech Republic, 2021.
32. Ahrenberg, L. Lines: An English-Swedish parallel treebank. In Proceedings of the 16th Nordic Conference of Computational Linguistics, Tartu, Estonia, 25–26 May 2007; pp. 270–273.
33. Conneau, A.; Rinott, R.; Lample, G.; Williams, A.; Bowman, S.; Schwenk, H.; Stoyanov, V. XNLI: Evaluating Cross-lingual Sentence Representations. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 2475–2485. [CrossRef]
34. Yang, Y.; Zhang, Y.; Tar, C.; Baldridge, J. PAWS-X: A Cross-lingual Adversarial Dataset for Paraphrase Identification. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Hong Kong, China, 3–7 November 2019; pp. 3687–3692. [CrossRef]
35. Zeman, D.; Hajič, J.; Popel, M.; Potthast, M.; Straka, M.; Ginter, F.; Nivre, J.; Petrov, S. CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. In Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Brussels, Belgium, 31 October–1 November 2018; pp. 1–21. [CrossRef]
36. Pan, X.; Zhang, B.; May, J.; Nothman, J.; Knight, K.; Ji, H. Cross-lingual Name Tagging and Linking for 282 Languages. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vancouver, BC, Canada, 30 July–4 August 2017; pp. 1946–1958. [CrossRef]
37. Artetxe, M.; Ruder, S.; Yogatama, D. On the Cross-lingual Transferability of Monolingual Representations. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 4623–4637. [CrossRef]
38. Clark, J.H.; Choi, E.; Collins, M.; Garrette, D.; Kwiatkowski, T.; Nikolaev, V.; Palomaki, J. TyDiQA: A benchmark for information-seeking question answering in typologically diverse languages. *Trans. Assoc. Comput. Linguist.* **2020**, *8*, 454–470. [CrossRef]
39. Brown, C.E., Coefficient of Variation. In *Applied Multivariate Statistics in Geohydrology and Related Sciences*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 155–157. [CrossRef]
40. Sutton, R.S.; McAllester, D.; Singh, S.; Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In Proceedings of the Advances in Neural Information Processing Systems, Denver, CO, USA, 29 November–4 December 1999; Volume 12.
41. Ghadimi, S.; Lan, G. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM J. Optim.* **2013**, *23*, 2341–2368. [CrossRef]

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