CoELM: Construction-Enhanced Language Modeling

Lvxiaowei Xu[1†], **Zhilin Gong**[1†], **Jianhua Dai**[2], **Tianxiang Wang**[1], **Ming Cai**[1*], **Jiawei Peng**[1]
[1]Department of Computer Science and Technology, Zhejiang University
[2]Zhejiang Institute of Administration
[1]{xlxw, zhilingong, cm}@zju.edu.cn [2]djh0803@163.com

Abstract

Recent studies have shown that integrating constructional information can improve the performance of pre-trained language models (PLMs) in natural language understanding. However, exploration into leveraging constructional information to enhance generative language models for natural language generation has been limited. Additionally, probing studies indicate that PLMs primarily grasp the syntactic structure of constructions but struggle to capture their semantics. In this work, we encode constructions as inductive biases to explicitly embed constructional semantics and guide the generation process. We begin by presenting a construction grammar induction framework designed to automatically identify constructions from corpora. Subsequently, we propose the Construction-Enhanced Language Model (CoELM). It introduces a construction-guided language modeling approach that employs a dynamic sequence reassembly strategy during pre-training. Extensive experiments have demonstrated the superiority of CoELM across various benchmarks.

1 Introduction

The constructions are linguistic structures that are viewed as a set of form-meaning pairs in Construction Grammar (CxG; Goldberg, 2003; Goldberg et al., 2005). Specifically, a construction is represented as a sequence of slot-constraints from a usage-based perspective (Dunn, 2017, 2019), where the slots encompass different levels of abstraction (e.g., lexical, syntactic). Based on these slots, constructions can be classified into three main types (Ungerer and Hartmann, 2023): (i) Schematic constructions contain open slots solely and convey a general meaning. For example, "Subject-Verb–Object1–Object2" is a ditransitive construction (Goldberg, 1995) that denotes the abstract meaning of transfer. (ii) Semi-idiomatic constructions are partially filled with lexical elements, for instance, "give-Pronoun-a-break". (iii) Idiomatic constructions are sequences composed of successive fixed words, such as "like a bat out of hell".

Recent studies (Tayyar Madabushi et al., 2020; Xu et al., 2023) have leveraged constructional information on encoder-only pre-trained language models (PLMs), such as BERT and RoBERTa (Devlin et al., 2019; Liu et al., 2019). These studies demonstrate that incorporating constructional information can enhance the performance of language models in various Natural Language Understanding (NLU) tasks. However, prior research has not explored the utilization of constructional information to augment generative language models for Natural Language Generation (NLG). In particular, the prevalent approach in generative models features a decoder-only architecture (OpenAI, 2023; Touvron et al., 2023), which contrasts with the encoder-only architecture in pre-training tasks. There is a significant gap that our work aims to address.

While PLMs can access constructional information without explicit training (Madabushi et al., 2023), probing studies (Weissweiler et al., 2023a,c; Tseng et al., 2022) suggest that they primarily grasp the syntactic structures but fail to fully capture semantic aspects. In Appendix B, we investigate the perception of language models for constructional semantics via probing experiments. Larger models exhibit better awareness of constructional semantics for both semi-idiomatic and schematic constructions compared to smaller ones, which aligns with the findings by Murty et al. (2023) on PLMs' capacity to generalize hierarchical structures, albeit slowly. These observations indicate that constructions consist of hierarchically abstract slots, posing intrinsic challenges for PLMs as they involve non-compositional meanings not directly linked to individual words. Consequently, this insight motivates us to adapt pre-training tasks to facilitate the swift

[†] Equal contribution and shared co-first authorship.
[*] Corresponding author.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 10061–10081

acquisition of constructional information.

In summary, our research focuses on developing effective pre-training strategies to efficiently learn constructional information, thereby enhancing language modeling capabilities essential for NLG. To this end, we propose a two-step approach. Initially, we employ automated methods to identify constructions from corpora, creating a comprehensive inventory for pre-training. Subsequently, these identified constructions are encoded as inductive biases during the pre-training phase, which allows model to explicitly capture constructional information and direct the generation process.

In the first step, recognizing the labor-intensive nature of manual construction inventory creation, we prioritize automated methods for identifying constructions. These methods are advantageous as they can uncover a variety of patterns that linguists might overlook (Madabushi et al., 2023). In usage-based CxG, a construction is depicted as a sequence of slot-constraints. To quantify the relationship between individual words and generate these constraints, either frequency-based or association-based models (Dunn, 2017) are applied. As suggested by Dunn (2019), association-based models provide better generalizations for slot-constraints when compared to frequency-based counterparts.

However, manual inspection of the inventory of constructions indicates that some constructions are relatively short or tend to contain generic labels (e.g., "Preposition + his"; Tsao and Wible, 2013; Tayyar Madabushi et al., 2020), lacking a clear mapping of form to a specific function or meaning. Through empirical study, we explore the possible reasons. Existing methods employ threshold-based strategies, such as frequency and association metrics (Dunn, 2017), to assess the eligibility of adding new slots to a construction. A construction is truncated if the association strength between adjacent slots drops below the specified threshold.

To tackle the issue, we introduce a construction grammar induction framework called CxGLearner. Within this framework, the association strength among slots is assessed through a PLM-based Association Strength Estimator (ASE), which can consider more extended distances when assessing slot constraints. Additionally, inspired by nucleus sampling (Holtzman et al., 2020), we establish a nucleus set that utilizes the ASE's output distribution. This set guides the decision on whether to append slots to the sequence of candidate constructions, aiming to create complete constructions without resorting to rigid threshold-based truncation.

For the second step, we propose a construction-guided language modeling approach with dynamic sequence reassembly strategy for pre-training. This strategy involves initially identifying all constructs within the input sequences that instantiate the constructions from the inventory. Then we insert constructions before their corresponding constructs, thereby explicitly embedding constructional semantics to guide the generation process. And the curriculum learning (Bengio et al., 2009) is employed to regulate the reassembly of both the types and quantities of constructions in sequences. Meanwhile, this approach ensures compatibility with existing generative language models, eliminating the need for additional decoding procedures.

Subsequently, our Construction-Enhanced Language Model (CoELM)[1] is pre-trained from scratch. Extensive experiments have validated the effectiveness of integrating constructional information into language modeling. Consequently, encoding constructions as inductive biases not only expedites the acquisition of constructional semantics for language model but also significantly enhances its performance across a range of benchmark tasks.

2 Induction Framework for CxG

In this section, we propose an unsupervised computational framework, which automatically inducts generic usage-based constructions from corpora.

2.1 Computational Language Framework

Previous works have attempted to automatically extract constructions from corpora (Dunn, 2017; Feng et al., 2022; Lyngfelt et al., 2018), with Dunn (2019) proposing a relatively sensible pipeline. However, as we mentioned in the Introduction, the search strategy in these works determines slot access constructions solely based on neighboring slots and a hard threshold, leading to truncation. To address this issue and ensure compatibility with language modeling, we propose a construction grammar induction framework (CxGLearner) to acquire an inventory of constructions. Since our framework comprises abundant implementation details, we briefly describe the entire pipeline in this section, leaving the detailed information to Appendix D.

As shown in Figure 1, our framework initially encodes the corpus into abstract representations

[1]Our code is publicly available at https://github.com/xlxwalex/CoELM

at different levels (e.g., lexical, syntactic). We then iterate through each position and level of slots among the encoded corpus, recursively searching to extract potential construction candidates. To determine whether slots can continue to expand potential candidate sequences, we propose the Association Strength Estimator (ASE), which is pre-trained to simulate the association strength between sequences and slots. Subsequently, these constructions are pruned and optimized (Appendix D) to acquire the final inventory of generic constructions.

2.2 Association Strength Estimator (ASE)

Since usage-based constructions are represented as sequences of slot constraints (Dunn, 2017), the key issue is how to measure the constraint relations between slots. Instead of frequency, association strength (Dunn, 2019; Gries, 2013) is applied to quantify the co-occurrence among multi-unit within a language. Prior research (Dunn, 2019) assesses if the association strength[2] between adjacent slots are sufficient for combination, which can only focus on localized features. As shown in Figure 1, a potential solution is to determine the association strength between the formed sequence of slots and the slot under consideration. However, this would introduce significant computational complexity for direct association strength assessment. Further details on complexity are discussed in Appendix A.

Recently, studies on the interpretability of language model backbones, specifically transformer, have shown that the attention mechanism in transformer exhibits a dynamic tendency to favor unique key tokens that often co-occur with query tokens (Tian et al., 2023a,b). This evidence aligns with the quantitative goal of measuring association strength, which motivates us to employ ASE, a pre-trained GPT-based language model, for estimating the association strength between sequences and slots.

We first split the background corpus for ASE pre-training. Then the pre-training task of ASE is adapted as next slot prediction, which randomly selects different abstraction levels of the slots in the background corpus. This facilitates the association between slots of varying levels. Then the probability distribution of the output can be exploited to estimate the association strength. The feasibility of ASE is also illustrated in Appendix H.

[2]Given that our candidate sequences initiate backward searches from arbitrary positions, and considering computational complexity, we opt to estimate unidirectional instead of bidirectional association strength of Dunn (2019).

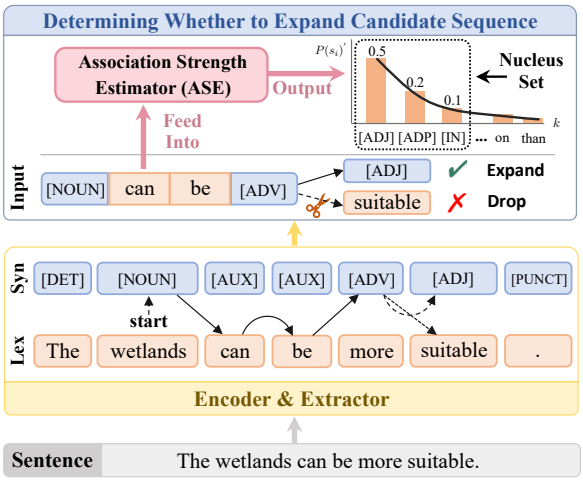

Figure 1: Schematic example of our framework.

2.3 Candidate Extraction with Nucleus Set

To extract construction candidates from the candidate corpus, we initiate a potential sequence with each slot in each sentence. We then conduct a recursive search for the next potential slot across different abstraction levels. A slot is incorporated into the current sequence if it is included in the nucleus set according to the output distribution of ASE. The inclusion process ceases when a slot no longer meets this criterion.

As shown in Figure 1, we regard the candidate sequence as the input for ASE. The ASE then generates a probability distribution for the next slot, denoted as $P(s_i)$. Drawing inspiration from nucleus sampling (Holtzman et al., 2020) and leveraging the unique aspects of ASE, we utilize the nucleus set instead of a hard threshold to prevent truncation.

We first rank the output probabilities from ASE in descending order. Subsequently, the top-k slots with the highest probability $P(s_i)$ are retained. Then the probabilities for these slots are normalized to acquire $P(s_i)'$. Finally, we select the smallest set of top slots, known as the nucleus set $N(p)$, whose cumulative probability surpasses the threshold p, that is, $\sum_{s \in N(p)} P(s_i)' \geq p$. The slot is deemed to fulfill the association condition if it is included in the nucleus set and can be expanded to the candidate sequence, otherwise, it is discarded. In Figure 1, a schematic example begins with the syntactic slot "NOUN" and forms a potential sequence of "NOUN-can-be-ADV". To determine whether the next possible slots, i.e. "ADJ" and "suitable", can be added to the sequence, we obtain the output distribution by feeding sequence into ASE. Then "ADJ" is included in the nucleus set

with a cumulative probability of 0.8 , while "suitable" does not meet the condition. Thus, "ADJ" can be expanded to the potential sequence.

This approach offers more flexibility in capturing the dynamic association strength between sequences and slots than the previous method.

3 Construction-Enhanced LM

In this section, we explore how the acquired construction inventory can be utilized to enhance the constructional awareness of language modeling.

3.1 Constructions as Inductive Bias

To employ constructions as inductive biases for steering the generative language model towards a more effective understanding of constructional semantics, we will address two main sub-issues:

(i) How to incorporate the meaning of construction to guide the generation? We propose a dynamic sequence reassembly strategy to tackle this issue. The input sequences are first matched for all the constructs and then distinguished by inserting corresponding constructions before the constructs into sequences. Thus, the constructional semantics can be explicitly fused to language model.

(ii) How to be compatible with existing language models, avoiding external decoding procedure, and adapting to various types of constructions at different granularities? For this issue, we introduce Curriculum Learning (Bengio et al., 2009) to dynamically regulate the selection of constructions.

3.2 Dynamic Sequence Reassembly Strategy

In order to leverage constructions, it is crucial to associate these patterns with their respective constructional semantics (i.e., meaning). We first integrate the construction inventory with the token vocabulary, embedding both into vector space by looking up the embedding matrix $E \in \mathbb{R}^{|\mathcal{V}|+|\mathcal{G}|}$, where $|\mathcal{V}|$ and $|\mathcal{G}|$ refer to the size of vocabulary and construction inventory, respectively. Then we enhance constructional perception in language models through dynamic sequence reassembly. As shown in Figure 2, we insert constructions before their corresponding constructs during pre-training, thus guiding the generation process. However, the complexity of matching multiple constructions from the encoded multi-level sequences arises due to the overlapping nature of usage-based CxG (Xu et al., 2023; Dunn, 2019). Thus, we refine the Aho-Corasick algorithm (Aho and Corasick, 1975) for construction matching, denoted as AC($\cdot$). It matches all

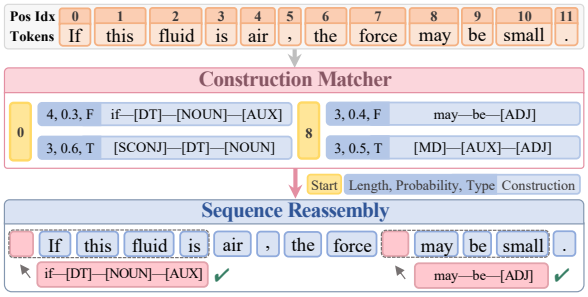

Figure 2: The schematic diagram of our CoELM. The type for *T* denotes *True*, while *F* stands for *False*.

constructions and generates a dictionary $\mathcal{M}$. This dictionary maps each position index of sequence to a set $\mathbf{m}$: the lengths $\mathbf{m}_L$, indexes $\mathbf{m}_I$, types $\mathbf{m}_T$ (true for schematic constructions, otherwise false) and probabilities $\mathbf{m}_P$ of all constructions beginning at that position. In the schematic sample of Figure 2, four constructions are matched at positions starting at 0 and 8. Then each position is traversed and the selection probability p_s is exploited to determine the insertion of constructions, while the abstract probability p_a is employed for dictating the likelihood of choosing schematic constructions. The CHOICE($\cdot$) function is designed to randomly select indexes for overlapping constructions according to a predefined probability distribution. In instances where no constructions are selected, the function yields a value of -1.

3.3 Architecture of Language Model

In this work, we pre-train a language model (CoELM) from scratch and our model architecture largely follows Llama 2 (Touvron et al., 2023), which aligns with the latest best practices in language modeling (Zhang et al., 2024): (1) We use Rotary Positional Embedding (RoPE; Su et al., 2024) as our preferred positional embedding. (2) To attain a more stable training, we normalize the input prior to each transformer layer with RMSNorm (Zhang and Sennrich, 2019). (3) Instead of the standard ReLU non-linearity, we adopt the approach of Llama 2 by combining Swish with the Gated Linear Unit, a method we refer to as SwiGLU (Shazeer, 2020). Additionally, Flash Attention (Dao, 2023) is employed to boost device throughput during the pre-training phase.

3.4 Training Method

In pre-training, our objective is to learn the relationship between constructions and the tokens that compose them. We approach the training process

Algorithm 1: Dynamic reassembly process

Input: Construction list $\mathcal{G}$ and encoded sequence $\mathcal{S}$. Probabilities of p_s and p_a.
Output: Reassembled sequence $\mathcal{S}^{'}$.

The length n and levels l of S: $n, l \leftarrow S$.SHAPE
The matching results $\mathbf{m} \in \mathcal{M} \leftarrow$ AC($\mathcal{S}$, $\mathcal{G}$, l, n)
Initialize $\mathcal{S}^{'} \leftarrow [\,]$
for $i = 0$ **to** $n - 1$ **do**
    token $\leftarrow \mathcal{S}[i, 0]$ // equivalent to lexical slot
    if *i* ***in*** $\mathcal{M}$.KEYS() **then**
        $\mathbf{m}_P^{'} \leftarrow [p \times p_a \times p_s$ **if** $\mathbf{m}_T$ **else** $p \times (1 - p_a) \times p_s$ **for** p **in** $\mathbf{m}_P]$
        ind $\leftarrow$ CHOICE($\mathbf{m}_I$, *weight*=$\mathbf{m}_P^{'}$)
        if *ind* $\geq$ *0* **then**
            token $\leftarrow \mathcal{S}[i : i + \mathbf{m}_L[\text{ind}]\,, 0]$
            $\mathcal{S}^{'} \leftarrow [\mathcal{S}^{'}, \mathbf{m}_I[\text{ind}]\,]$
            $i \leftarrow i + \mathbf{m}_L[\text{ind}]$
        end
    end
    $\mathcal{S}^{'} \leftarrow [\mathcal{S}^{'}, \text{token}]$
end
return $\mathcal{S}^{'}$

from two perspectives: (1) Schematic constructions have more abstract slots than semi-idiomatic and idiomatic constructions, resulting in relatively coarser constructional semantics. Thus, we anticipate the model to distinguish schematic constructions during the early stage, and focus more on semi-idiomatic and idiomatic constructions later. (2) Since the language model can learn hierarchical structures, albeit slowly, we only guide the generation process at an early stage. This strategy not only expedites learning but also eliminates the need for specialized handling of the construction units in the generated sequence during inference.

Drawing on the approach of Bengio et al. (2009), we adopt curriculum learning to achieve our objectives. They are both employed in the dynamic reassembling process (p_s and p_a) in Algorithm 1. Curriculum learning defines the pacing function that regulates the sequence and timing of introducing various difficulty levels of samples during training. As depicted in Figure 3, we establish two distinct pacing functions for different aspects of our goal. The model initiates with a predetermined number of warm-up steps t_w to establish token relationship. This is succeeded by a stage where the construction guidance is progressively diminished until stop steps t_s. For the selection and abstraction probabilities, we employ linear-decay and cosine-decay pacing functions, respectively.

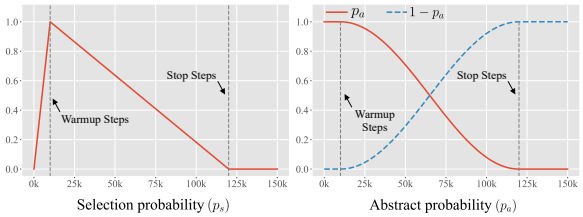

Figure 3: The pacing functions of selection probability p_s and abstract probability p_a over pre-training process.

4 Experiments

4.1 Experiments Setup

Pre-training corpus. The entire pre-training corpus is sampled from multiple subsets of RedPajama (Soboleva et al., 2023) and Pile (Gao et al., 2020), which aligns with the settings in previous pre-trained language models. Our corpus comprises a total of 188B tokens after pre-processing (Chen et al., 2023), such as deduplication. The detailed statistics and pre-processing procedure of the pre-training corpus are provided in Appendix C.

Evaluation tasks. For evaluating the performance of model (CoELM), we utilize the Language Model Evaluation Harness (Gao et al., 2023) framework to conduct evaluation on various language modeling and commonsense reasoning tasks.

Implementation. In CxGLearner framework, we set the parameters of k and p to be 20 and 0.6, respectively. In total, we induct 13,262 generic constructions for the inventory. As for our CoELM, we adopt the similar hyper-parameters in Pythia-410M (Biderman et al., 2023) for comparison. This includes 24 layers with 16 heads, with a hidden size of 1024 and an intermediate size set to 4096. We employ the SentencePiece (Kudo and Richardson, 2018) as the tokenizer with the vocab size to be 50257. During pre-training, we utilize the AdamW optimizer (Loshchilov and Hutter, 2018) with a cosine learning rate scheduler. We exploit 2,000 warmup steps and the batch size is set to 1 million tokens. We pre-train 250,000 steps for running 1.4 epoch on entire corpus. And the t_w and t_s for curriculum learning are set to 2,000 and 80,000 steps, respectively. More detailed information for settings is demonstrated in Appendix F.

Comparable LMs. We primarily focus on language models which are close to the number of parameters in our implementation. Specifically, we compare our CoELM with GPT-2 Medium (345M; Radford et al., 2019), OPT-350M (Zhang et al.,

Model	Arc-E	Arc-C	OBQA	Logi.	PIQA	Hella.	BoolQ	WSC	QNLI	Avg.
Zero-shot Results										
GPT-2	49.0	21.6	18.6	22.4	67.6	33.3	58.6	40.4	49.4	40.1
OPT	44.0	20.7	17.6	21.0	64.4	32.0	57.7	36.5	49.5	38.2
Pythia	51.3	20.5	17.6	**23.8**	67.2	34.5	**58.2**	51.0	50.1	41.6
BLOOM	47.4	22.4	17.2	22.6	64.0	31.6	55.1	40.4	**50.3**	39.0
CoELM	**55.9**	**24.6**	**21.2**	21.0	**70.9**	**40.3**	56.4	**55.8**	50.2	**44.0**
Five-shot Results										
GPT-2	50.9	22.8	19.2	22.7	66.3	32.9	61.4	36.5	48.8	40.2
OPT	45.5	20.6	18.4	23.2	65.6	31.9	57.2	**49.0**	51.1	40.3
Pythia	53.5	22.1	18.0	**23.8**	67.7	34.6	59.3	47.1	**50.6**	41.9
BLOOM	50.3	21.6	17.4	21.4	64.2	31.5	**59.4**	40.4	49.5	39.5
CoELM	**59.1**	**25.7**	**21.4**	20.3	**70.7**	**37.3**	57.7	38.5	49.5	**42.4**

Table 1: Five-shot and zero-shot results on NLP Benchmarks. The metric of these tasks is accuracy and the best result on each task is in **bold**. The "**Logi.**" task denotes LogiQA, while "**Hella.**" task stands for HellaSwag.

2022), Pythia-410M (Biderman et al., 2023) and BLOOM-560M (Workshop et al., 2022).

4.2 Experiment Results

To evaluate the ability of CoELM for reasoning and language modeling, we conduct experimental comparisons on several common benchmarks (Statistics for these tasks are shown in Appendix E). Our main results on the comprehension and reasoning tasks are shown in Table 1. We conduct both zero-shot and few-shot experiments for comparison, from which several observations can be obtained.

First, Pythia outperforms other language models under zero-shot setting, while OPT and BLOOM exhibit similar performance, despite OPT having fewer parameters. GPT-2 achieves higher performance than OPT and BLOOM, but lower than Pythia. Remarkably, our CoELM significantly outperforms other models on most of tasks, especially with huge gaps in the Arc-E, HellaSwag and WSC tasks. This suggests that our CoELM, with constructional information incorporated, possesses superior comprehension and reasoning capabilities.

Second, few-shot learning is designed to answer the questions generalized from a small number of examples. Thus, five exemplars are provided for the prompts. We observe that the performance of language models can be enhanced in the majority of tasks through a few-sample setting, although on a few tasks, the performance decreases instead. For instance, in HellaSwag, a natural language reasoning task with complex contextual premises, a deeper understanding of linguistic knowledge inherent in the language models is required. Consequently, irrelevant knowledge from other exemplars may interfere with answer generation.

Model	Zero-shot		Five-shot	
	PPL.	Acc.	PPL.	Acc.
GPT-2	18.3	43.1	35.1	31.9
OPT	16.4	45.1	23.9	38.3
Pythia	10.5	52.5	**15.1**	**44.8**
BLOOM	28.7	35.3	41.6	30.2
CoELM	**9.3**	**53.7**	28.1	36.4

Table 2: Five-shot and zero-shot results on LAMBADA task."**PPL.**" and "**Acc.**" stands for perplexity and accuracy metrics, respectively.

Third, the OPT model has the largest improvement in few-shot learning compared to the zero-shot setting. Additionally, all models show significant gains on the OBQA task and our CoELM also has a large improvement on the Arc-E task. Although the average performance of our CoELM is reduced compared to the zero-shot setting due to the effect of the WSC task, CoELM achieves higher average performances across all the benchmarks than other language models.

Moreover, we evaluate the performance on generative task for word prediction. The main results of LAMBADA task are demonstrated in Table 2. In the zero-shot setting, it can be observed that Pythia is significantly better than other language models. And our CoELM outperforms Pythia, demonstrating its language modeling capabilities. However, the performance of all the language models is significantly lower in the few-shot setting compared to zero-shot, which can be due to the effect of ir-

Model	Arc-E	Arc-C	PIQA	WSC
Pythia	52.57	20.90	67.03	**62.50**
CoELM	**53.83**	**24.57**	**69.59**	59.62
w/o DSR	51.01	22.78	68.82	36.54

Table 3: Experimental results of ablation study.

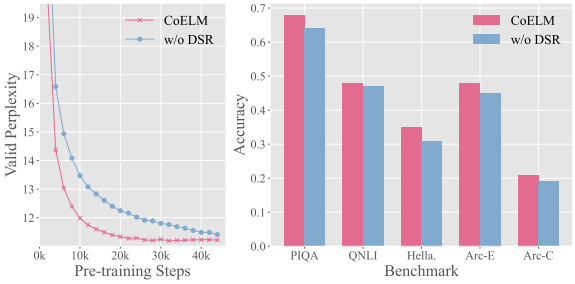

Figure 4: Valid perplexity curve and performance comparison during pre-training of language modeling.

relevant context on the accuracy of the language modeling as well as the ability of the language models to generate long texts. In few-shot setting, Pythia is the least affected and achieves the best performance. The results also suggest the necessity for our CoELM to explore the pre-training process in the long-text scenarios in the future.

These observations confirm the validity of CoELM, with its construction-enhanced generation process. Furthermore, constructions can enhance language modeling across these benchmarks.

4.3 Comparative Analysis

Ablation Study. Since the language models are pre-trained on different sizes and types of corpora as well as varying hyper-parameter settings, we conduct ablation study experiments to further investigate the effectiveness of our CoELM. Therefore, we pre-train a language model from scratch based on Llama architecture, same as our CoELM. We ensure that all hyper-parameters are consistent with our CoELM, except that the model will not be applied to dynamic sequence reassembly (DSR). This model is referred to as w/o DSR. Our model is pre-trained on corpora with 188B tokens, which is smaller than other pre-trained models. For example, Pythia is pre-trained on 207B corpora. As the Pythia suite provides the intermediate results during pre-training, we employ the results of Pythia with 93k steps, which is pre-trained an entire epoch on the corpora. Similarly, we use the intermediate results of the Llama baseline model and CoELM pre-trained on a single epoch for comparison. The results on four benchmarks are shown in Table 3.

It can be observed that the performance of Pythia is slightly lower than that demonstrated in Table 1, which suggests that more pre-training steps could potentially enhance the performance of PLMs. Meanwhile, Pythia and w/o DSR perform very close to each other except for WSC task, indicating that the differences in corpora have less impact on language modeling. Our CoELM achieves better performances on most tasks, further supporting the validity of constructional incorporation for enhancing language modeling capabilities.

Language Modeling Acceleration. During pre-training, the curves of loss and perplexity of language models first plummet in a power law, and then seemingly enter a near-linear decline. As the constructions are employed as the inductive bias for our CoELM, they guide the generation process. We observe that they can also accelerate the convergence of the model during the pre-training process, as shown in Figure 4. The CoELM w/o DSR enters a linear descent phase at around 40k steps, while CoELM requires only 15k steps and exhibits lower perplexity. Thus, at the end of pre-training, our CoELM also has a lower perplexity on the validation corpus with respect to w/o DSR.

Furthermore, we compare the intermediate performances on five benchmarks between CoELM with 20k steps and w/o DSR with 40k steps and the results are illustrated in Figure 4. In all five tasks, CoELM outperforms w/o DSR. Particularly, there is a big gap in the HellaSwag task. This demonstrates that with the assistance of constructional information, the language model can establish relationships among tokens more efficiently, thus accelerating the language modeling procedure. This characteristic may be extendable to larger scale language models, left for future exploration.

Wug Test. Prior work (Weissweiler et al., 2023b) has applied the wug test, a probing task used to investigate whether the pre-trained language models possess human-like linguistic knowledge. The wug test focuses on morphology, creating non-existent words according to systematic patterns of covariation in form and meaning, such as the past tense variation. This task contains 50 irregular verbs and perturbs these verbs by one or two letters to produce new words. Then these verbs are annotated for past tense of the nonce word, e.g., , *wug* $\rightarrow$ *wugged*. Meanwhile, annotators are requested to

N-shot	Model	PPL.	Accuracy
Zero	Pythia-410M	4137.4	0.04
	Pythia-1B	1323.3	0.04
	TinyLlama	**842.7**	0.08
	CoELM	893.4	**0.14**
Five	Pythia-410M	48.8	0.22
	Pythia-1B	**11.4**	**0.40**
	TinyLlama	11.7	0.34
	CoELM	18.3	0.36

Table 4: The zero-shot and few-shot experimental results of wug test. "**PPL.**" stands for perplexity metric. The lowest perplexity and highest accuracy is in **bold**.

provide multiple possible past tense forms based on morphology for these verbs.

Therefore, we investigate whether constructions can contribute to addressing unseen words based on the linguistic knowledge for language models. To evaluate the performance of wug test, we follow Weissweiler et al. (2023b), considering an instance correct only if the generated answer falls within the set of possible past tense forms of the instance. We report the lowest perplexity of the possible forms within the set and the accuracy in Table 4. The 410M and 1B size models of Pythia and TinyLlama (Zhang et al., 2024) with 1.1B parameters are employed for the comparison. For TinyLlama, we utilize the intermediate model that is pre-trained with 240k steps and fed to 500B tokens.

In zero-shot setting, our CoELM significantly outperforms the other language models, despite both Pythia-1B and TinyLlama having a far greater number of parameters. This indicates that our CoELM has a better comprehension of linguistic knowledge by incorporating constructional information, enabling it to generalize to unseen words. As for the few-shot learning[3], Pythia-1B and TinyLlama achieve better performances on wug test. This is due to the better in-context learning capability of the larger scale models (Dong et al., 2022). However, CoELM also improves with few-shot prompting and outperforms Pythia-410M, demonstrating the effectiveness of our CoELM.

Dimensionality Reduction Analysis. To further explore the impact of constructional information for language modeling, we conduct a visual analysis of representation obtained from our CoELM, w/o DSR and Pythia. First, we randomly select five semi-idiomatic constructions from the learned inventory. Then we search the WikiText corpus (Merity et al., 2017) for 50 constructs of each construction. These constructs are fed into different models and the representations are acquired from the last token of each construct. Subsequently, we apply dimensionality reduction using UMAP (McInnes et al., 2018). As shown in Figure 5, the representations in CoELM form clusters with distinct boundaries, while the representations in w/o DSR and Pythia are diffuse among different clusters. This observation suggests that our CoELM effectively incorporates constructional semantics through the dynamic sequence reassembly strategy.

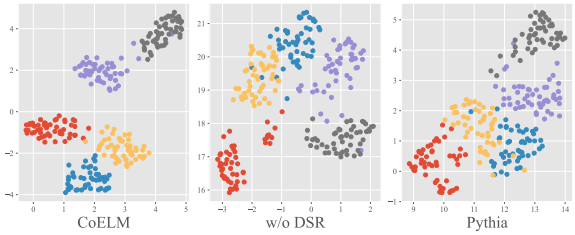

Figure 5: 2-D UMAP plot for construct representation.

5 Related Work

Construction Grammar (CxG) has been applied to various natural language processing tasks. Tsao and Wible (2013) utilize constructions as contextual features for word similarity detection. Raghuram et al. (2017) leverage Embodied CxG to address the reference resolution problem, and Nevens et al. (2019) apply Fluid CxG for semantic parsing in visual question answering. Xu et al. (2023) integrate constructional information to improve the performance of PLMs in NLU tasks. However, there has been no effort to ascertain whether constructions can provide benefits for generative language models. Our work aims to bridge this gap.

Recent research investigates whether PLMs can access constructions. Tayyar Madabushi et al. (2020) find that BERT is capable of accessing a considerable amount of constructional information without explicit training. Li et al. (2022) conduct probing studies to show that PLMs can capture argument structure constructions. Tseng et al. (2022) investigate PLMs' awareness of constructions and find that predicting open slots is more challenging than closed ones. Weissweiler et al. (2022) find that while PLMs can recognize the construction, they struggle with understanding its meaning. Fur-

[3]For few-shot setting, we do not employ the prompts from Language Model Evaluation Harness framework in addition to the other Baselines from CoELM since the prompts constructed from the wug dataset introduce more unseen words, see Appendix J for more detailed discussion.

thermore, Weissweiler et al. (2023c) discover that both autoregressive and non-autoregressive PLMs can distinguish constructional patterns but fail to apply their meanings. These findings underscore the need to adapt pre-training tasks to enable PLMs to rapidly acquire constructional information.

An inventory of constructions is a valuable resource for construction-based research. As manual extraction is often time consuming, automated approaches have been proposed to identify constructions from corpora. These methods are advantageous for uncovering patterns that may have been previously unexplored or highly conventional (Madabushi et al., 2023), potentially overlooked in traditional analyses. Wible and Tsao (2010) present a hybrid n-grams model to discover constructions that meet a frequency threshold. Dunn (2017) proposes an induction algorithm to extract constructions based on association strength. Lyngfelt et al. (2018) develop automatic tools to detect partially schematic constructions using frequency metrics. Dunn (2019) suggests that association-based models produce better generalizations for slot-constraints compared to frequency-based models. However, manual analyses reveal that some extracted constructions are short or lack informativeness (Lyngfelt et al., 2018; Tayyar Madabushi et al., 2020). Li et al. (2022) argue that there is currently a lack of a high-quality and wide-coverage construction dataset. Therefore, we investigate potential reasons and develop a construction induction framework to address the issue.

6 Conclusion

Our research is dedicated to developing effective pre-training strategies for the rapid acquisition of constructional information to enhance language modeling capabilities. To achieve this goal, we initially introduce a construction induction framework that automatically identifies constructions from corpora. We employ a PLM-based Association Strength Estimator (ASE) and a nucleus set to generate complete constructions, avoiding the rigid threshold-based truncation. Subsequently, we propose a construction-guided language modeling approach utilizing a dynamic sequence reassembly strategy. This method encodes constructions as inductive biases during pre-training, thereby guiding the generation process. We pre-train our Construction-Enhanced Language Model (CoELM) from scratch, and extensive experiments have confirmed the effectiveness of CoELM.

Limitations

In this work, the limitations can be summarized into two main aspects for future discussion:

(1) As we mentioned in Appendix C, we primarily focus on the generation of natural language from a linguistic perspective, without utilizing code data for pre-training procudure. However, constructions also exist in code (Mosses, 2021), which can be viewed as pairing of form and function. Nevertheless, they differ in their presentation compared to constructions in linguistics. Therefore, we will further discuss them in future work.

(2) In our main experiments, we observe that smaller-scale models exhibit diminished performance on instruction following and long text, resulting in insignificant improvements over the zero-shot setting. Therefore, we will further explore the refinement of the pre-training data procedure and the optimization of model architecture.

(3) In order to acquire better interpretable constructions, as we stated in Appendix D.1, we utilize the language-specific hierarchy, i.e., XPOS. Consequently, our pre-training procedure is conducted exclusively with the English corpus (More details are shown in Appendix C). This limitation restricts our CoELM from being multilingual. In subsequent research endeavors, we intend to investigate the significance of generic constructions within the domain of language modeling.

(4) Due to limited computational resources, we are constrained to pre-training a relatively small-scale model. However, the probing experiments we conduct in Appendix B reveal that incorporating constructional information into larger-scale models may also enhance their language modeling capabilities, which we leave for future exploration.

Ethics Statement

In this work, we use publicly available corpora and benchmarks under their licenses. These publicly available data are checked to ensure that they do not include any offensive and illegal content.

Acknowledgements

We thank all anonymous reviewers for their insightful comments and suggestions. This research is supported by the Science and Technology Project of Zhejiang Province (2022C01044) and the National Natural Science Foundation of China (51775496).

References

Alfred V Aho and Margaret J Corasick. 1975. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

Daoyuan Chen, Yilun Huang, Zhijian Ma, Hesen Chen, Xuchen Pan, Ce Ge, Dawei Gao, Yuexiang Xie, Zhaoyang Liu, Jinyang Gao, et al. 2023. Data-juicer: A one-stop data processing system for large language models. *arXiv preprint arXiv:2309.02033*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, and Ziqing Yang. 2021. Pre-training with whole word masking for chinese bert. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3504–3514.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets. *arXiv preprint arXiv:1809.02922*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Jonathan Dunn. 2017. Computational learning of construction grammars. *Language and cognition*, 9(2):254–292.

Jonathan Dunn. 2019. Frequency vs. association for constraint selection in usage-based construction grammar. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 117–128, Minneapolis, Minnesota. Association for Computational Linguistics.

Rui Feng, Congcong Yang, and Yunhua Qu. 2022. A word embedding model for analyzing patterns and their distributional semantics. *Journal of Quantitative Linguistics*, 29(1):80–105.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Adele Goldberg. 1995. Constructions: a construction grammar approach to argument structure. *Chicago: The University of Chicago*.

Adele Goldberg. 2006. Constructions at work: Constructionist approaches in context. *Oxford University Press on Demand*.

Adele E Goldberg. 2003. Constructions: A new theoretical approach to language. *Trends in cognitive sciences*, 7(5):219–224.

Adele E Goldberg, Mirjam Fried, and Jan-Ola Östman. 2005. *Construction grammar (s): Cognitive and cross-language dimension*. John Benjamins Amsterdam.

John Goldsmith. 2006. An algorithm for the unsupervised learning of morphology. *Natural language engineering*, 12(4):353–371.

Stefan Th Gries. 2013. 50-something years of work on collocations: What is or should be next. *International Journal of Corpus Linguistics*, 18(1):137–166.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. 2022. Training compute-optimal large language models. In *Proceedings of NeurIPS*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification*, 2:193–218.

Ray Jackendoff. 2003. *Foundations of Language*. Oxford University Press.

Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. 2023. Scaling sentence embeddings with large language models. *arXiv preprint arXiv:2307.16645*.

Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. 1983. Optimization by simulated annealing. *science*, 220(4598):671–680.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71. Association for Computational Linguistics.

Ronald W Langacker. 1987. *Foundations of cognitive grammar: Theoretical prerequisites*, volume 1. Stanford university press.

Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. 2022. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445. Association for Computational Linguistics.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*.

Bai Li, Zining Zhu, Guillaume Thomas, Frank Rudzicz, and Yang Xu. 2022. Neural reality of argument structure constructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7410–7423. Association for Computational Linguistics.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3622–3628. International Joint Conferences on Artificial Intelligence Organization. Main track.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*.

Benjamin Lyngfelt, Linnéa Bäckström, Lars Borin, Anna Ehrlemark, and Rudolf Rydstedt. 2018. Constructicography at work: Theory meets practice in the swedish constructicon. In *Constructicography*, pages 41–106. John Benjamins.

Harish Tayyar Madabushi, Laurence Romain, Petar Milin, and Dagmar Divjak. 2023. Construction grammar and language models. *arXiv preprint arXiv:2308.13315*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60. Association for Computational Linguistics.

Christopher D Manning. 2011. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In

International conference on intelligent text processing and computational linguistics, pages 171–189. Springer.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Leland McInnes, John Healy, and Steve Astels. 2017. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. 2018. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391. Association for Computational Linguistics.

Peter D Mosses. 2021. Fundamental constructs in programming languages. In *Leveraging Applications of Formal Methods, Verification and Validation: 10th International Symposium on Leveraging Applications of Formal Methods, ISoLA 2021, Rhodes, Greece, October 17–29, 2021, Proceedings*, volume 13036, page 296. Springer Nature.

Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher Manning. 2023. Grokking of hierarchical structure in vanilla transformers. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 439–448. Association for Computational Linguistics.

Jens Nevens, Paul Van Eecke, and Katrien Beuls. 2019. Computational construction grammar for visual question answering. *Linguistics Vanguard*, 5(1).

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534. Association for Computational Linguistics.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *LREC'12*, pages 2089–2096. European Language Resources Association.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Vivek Raghuram, Sean Trott, Kelly Shen, Ethan Goldberg, and Sidney Oderberg. 2017. Semantically-driven coreference resolution with embodied construction grammar. In *2017 AAAI Spring Symposium Series*.

Giulia Rambelli, Emmanuele Chersoni, Philippe Blache, Chu-Ren Huang, and Alessandro Lenci. 2019. Distributional semantics meets construction grammar. towards a unified usage-based model of grammar and meaning. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 110–120. Association for Computational Linguistics.

Jorma Rissanen. 1978. Modeling by shortest data description. *Automatica*, 14(5):465–471.

Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama.

Luc Steels and Joachim de Beule. 2006. A (very) brief introduction to fluid construction grammar. In *Proceedings of the Third Workshop on Scalable Natural Language Understanding*, pages 73–80. Association for Computational Linguistics.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Harish Tayyar Madabushi, Laurence Romain, Dagmar Divjak, and Petar Milin. 2020. CxGBERT: BERT meets construction grammar. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4020–4032. International Committee on Computational Linguistics.

Yuandong Tian, Yiping Wang, Beidi Chen, and Simon Shaolei Du. 2023a. Scan and snap: Understanding training dynamics and token composition in 1-layer transformer. In *Conference on Neural Information Processing Systems*.

Yuandong Tian, Yiping Wang, Zhenyu Zhang, Beidi Chen, and Simon Du. 2023b. JoMA: Demystifying multilayer transformers via JOint dynamics of MLP and attention. In *NeurIPS 2023 Workshop on Mathematics of Modern Machine Learning*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Nai-Lung Tsao and David Wible. 2013. Word similarity using constructions as contextual features. In *Proceedings of the Joint Symposium on Semantic Processing. Textual Inference and Structures in Corpora*, pages 51–59, Trento, Italy.

Yu-Hsiang Tseng, Cing-Fang Shih, Pin-Er Chen, Hsin-Yu Chou, Mao-Chang Ku, and Shu-Kai Hsieh. 2022. CxLM: A construction and context-aware language model. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 6361–6369. European Language Resources Association.

Tobias Ungerer and Stefan Hartmann. 2023. *Constructionist approaches: Past, present, future*. Elements in Construction Grammar. Cambridge University Press.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Leonie Weissweiler, Taiqi He, Naoki Otani, David R. Mortensen, Lori Levin, and Hinrich Schütze. 2023a. Construction grammar provides unique insight into neural language models. In *Proceedings of the First International Workshop on Construction Grammars and NLP (CxGs+NLP, GURT/SyntaxFest 2023)*, pages 85–95. Association for Computational Linguistics.

Leonie Weissweiler, Valentin Hofmann, Anjali Kantharuban, Anna Cai, Ritam Dutt, Amey Hengle, Anubha Kabra, Atharva Kulkarni, Abhishek Vijayakumar, Haofei Yu, Hinrich Schuetze, Kemal Oflazer, and David Mortensen. 2023b. Counting the bugs in ChatGPT's wugs: A multilingual investigation into the morphological capabilities of a large language model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6508–6524. Association for Computational Linguistics.

Leonie Weissweiler, Valentin Hofmann, Abdullatif Köksal, and Hinrich Schütze. 2022. The better your syntax, the better your semantics? probing pretrained language models for the English comparative correlative. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10859–10882. Association for Computational Linguistics.

Leonie Weissweiler, Valentin Hofmann, Abdullatif Köksal, and Hinrich Schütze. 2023c. Explaining pretrained language models' understanding of linguistic structures using construction grammar. *Frontiers in Artificial Intelligence*, 6.

Aaron Steven White, Pushpendre Rastogi, Kevin Duh, and Benjamin Van Durme. 2017. Inference is everything: Recasting semantic resources into a unified evaluation framework. In *IJCNLP (Volume 1: Long Papers)*, pages 996–1005. Asian Federation of Natural Language Processing.

David Wible and Nai-Lung Tsao. 2010. StringNet as a computational resource for discovering and investigating linguistic constructions. In *Proceedings of the NAACL HLT Workshop on Extracting and Using Constructions in Computational Linguistics*, pages 25–31, Los Angeles, California. Association for Computational Linguistics.

BigScience Workshop, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Lvxiaowei Xu, Jianwang Wu, Jiawei Peng, Zhilin Gong, Ming Cai, and Tianxiang Wang. 2023. Enhancing language representation with constructional information for natural language understanding. In *ACL*, pages 4685–4705. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800. Association for Computational Linguistics.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhi-Hua Zhou. 2012. *Ensemble methods: foundations and algorithms*. CRC press.

A Background of Construction Grammar

Construction Grammar (CxG) is an influential paradigm within cognitive linguistics, positing that grammar encompasses a meaningful continuum of lexicon, morphology, and syntax. This perspective diverges from traditional views that prioritize a stable framework of arbitrary rules for producing well-formed sequences. In CxG, the fundamental units of grammar are constructions, which is defined as symbolic pairings of form and meaning (Langacker, 1987; Goldberg, 1995, 2006). These pairings underscore the central thesis of the theory: grammatical structure is inherently meaningful, with each construction serving as a vital component in the conveyance of specific semantic content. This approach emphasizes their role in linking morphosyntactic patterns directly to their associated meanings. As described in the Introduction, Construction Grammar (CxG) is characterized by its diverse syntactic structures, which span different levels of abstraction. These structures can comprise both partially and fully specified components (Goldberg, 2003), which allows for the substitution of specific words for productivity.

The CxG paradigm has evolved to several distinct implementations, each adopting a unique perspective on language structure and processing. These include the formal approaches such as Fluid Construction Grammar (FCG; Steels and de Beule, 2006), which emphasizes the dynamic nature of linguistic constructions, while Embodied Construction Grammar (ECG; Goldberg et al., 2005) integrates the role of sensory and motor experiences in linguistic interpretation. Sign-Based Construction Grammar (SBCG; Rambelli et al., 2019) combines insights from formal linguistic theory with sign-based language analysis. However, these formal approaches all rely heavily on manual definition with prior knowledge, such as FrameNet (Baker et al., 1998). In usage-based CxG, patterns that occur with sufficient frequency can also be considered as constructions (Goldberg, 2006). It enables constructions to be acquired from the corpus.

In previous research, Dunn (2017, 2019) propose a computationally slot-constraints pipeline for construction extraction, which is composed of three stages: (1) In representation stage, the slots are represented to three levels, i.e., lexical, syntactic and joint semantic-syntactic. The syntactic slots are denoted to the universal POS tags, while the semantic-syntactic slots are formed from static embedding of words (e.g., FastText) that are clustered in discrete semantic domains via K-Means clustering. (2) In construction identification stage, Dunn (2017, 2019) initially measures the slot constraints based on the association strength, which is applied to quantify the co-occurrence among multi-units within a language (Gries, 2013). The unidirectional association strength Δ_u can be defined as:

$$\Delta_u = \frac{C_{(x,y)}}{C_x} - \frac{C_y - C_{(x,y)}}{C_{\text{all}} - C_x} \quad (1)$$

where x and y are the adjacent slots. C_x and C_y are the frequency of x and y in the corpus, while $C_{(x,y)}$ is the frequency of co-occurrence of x and y. And C_{all} is the total number of occurrences of all slots. In order to compute the association strength, Dunn (2017, 2019) first counts the corpus and calculates the frequency of 1-gram and 2-gram items. However, this approach can only assess the association strength between two adjacent slots, thus focusing solely on localized features. Following this, candidate sequences are generated through a recursive search conditioned on the association strength between slots using a hard threshold. Subsequently, these candidate sequences are pruned to derive potential constructions. (3) In candidate evaluation stage, the tabu search is applied to determine the optimal set of constructions with Minimum Description Length (MDL; Rissanen, 1978) metric. Then the inventory of constructions is acquired.

In order to assess slot by considering longer slot constraints for construction candidates, one possible solution would be to determine the association strength between the formed sequence of slots and the slot under consideration. However, if we directly compute the association strength between sequences and slots based on the corpus via the approach of Dunn (2019), this would introduce an unacceptable computational complexity. As an example, if we have N possible slots and the currently formed sequence is of length L, then in order to assess the association strength between the sequence and the $(L+1)$-th slot, we have two implementations: (1) we can store all the candidates of length L. (2) The frequency of occurrences of sequences and slots can be re-computed in the corpus each time. For approach (1), it would require storing N^L floating point numbers for the frequency of L-gram items in the worst case. If both N and L are large (In practice, N is about 50k, and L is between 3 and 7), it may exceed the server's capacity. As for approach (2), it is computationally infeasi-

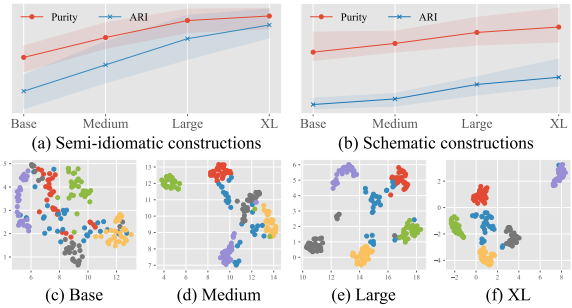

Figure 6: The probing experiments of constructional semantics among different scales of GPT-2 models.

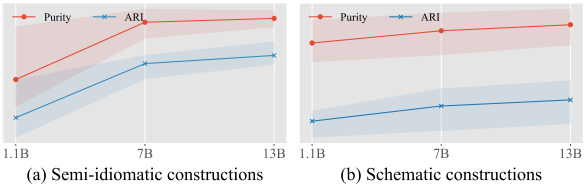

Figure 7: The probing experiments of constructional semantics among different scales of Llama models.

ble to re-compute the frequency of sequences each time from a corpus containing around 3B tokens. Therefore, we propose the Association Strength Estimator (ASE) to approximate the association strength between sequences and slots, which can compute the association strength fast on a limited memory footprint.

B Probing Experiments

To investigate the ability of PLMs to capture constructional semantics, we conduct probing experiments on different scales of GPT-2 language models (Radford et al., 2019). We focus primarily on semi-idiomatic and schematic constructions, which exhibit productivity and can be filled with words in their abstraction slots. As shown in Figure 6, we randomly select five semi-idiomatic and schematic constructions, respectively. Then we match 50 constructs (abstract slots filled with actual words) from WikiText corpus (Merity et al., 2017) for each construction. After feeding these constructs into the GPT-2 models to obtain representations (Jiang et al., 2023), we apply dimensionality reduction with UMAP (McInnes et al., 2018) and then cluster the reduced representation via HDBCAN (McInnes et al., 2017). As these constructs are the instantiation of constructions, their representations also inherit the constructional semantics of corresponding constructions (Goldberg, 2003). For constructs instantiated from the same construction, it should be more plausible that they will be clustered together. Thus, we utilize two metrics for evaluating clustering accuracy: Purity and the Adjusted Rand Index (ARI; Hubert and Arabie, 1985). The higher these metrics are, the better the model can distinguish the constructional semantics.

We perform five independent probing experiments, which are shown in Figure 6 (a) and (b). Our main finding is that larger models indeed exhibit constructional semantic awareness capabilities for both semi-idiomatic and schematic constructions. This aligns with the idea that larger models, trained with larger corpora and more training steps under the Scaling Law (Hoffmann et al., 2022), are more adept at capturing such nuances. Meanwhile, (c)~(f) are the clustering results after dimensionality reduction. It is observed that the representations in larger models form clusters with distinct boundaries, while the representations in smaller models are diffuse among different clusters

To investigate if the observed trend in constructional semantic perception on larger scale models aligns with our findings, we then conduct probing experiments on Llama models (Touvron et al., 2023). As shown in Figure 7, we exploit TinyLlama-1.1B, Llama-7B, and Llama-13B for our experiment. TinyLlama (Zhang et al., 2024) is pre-trained on SlimPajama (Soboleva et al., 2023), which replicates the pre-training data of Llama, and thus can be directly compared with the other two Llama models. In terms of trends, the Llama series with a larger number of parameters exhibits the same behavior as GPT-2 series. In both semi-idiomatic and schematic constructions, we observe that the purity and ARI metrics increase with the number of parameters. This also indicates that larger models indeed have more powerful modeling capabilities for constructional semantics. Our findings suggest the potential of regarding constructions as the inductive bias for language models to boost their performance.

C The Details of Our Pre-training Data

Our training corpus comprises a variety of sources, which cover a diverse set of domains. They are mainly from RedPajama (Soboleva et al., 2023) and Pile (Gao et al., 2020), which align with previous practices in pre-training language models. As we discuss in Limitations, our work is centered on the generation of natural language, whereas code behaves differently in terms of constructions. Con-

Corpus	Tokens	Sampling prop.	Epochs
CCNews	63.7 B	30.5%	1.24
C4	66.0 B	31.3%	1.23
Wikipedia	3.3 B	3.5%	2.76
Books	21.0 B	15.0%	1.86
Law	10.2B	5.7%	1.45
ArXiv	19.3 B	11.0%	1.48
USPTO	3.5 B	2.5%	1.86
Other	1.2 B	0.5%	1.08

Table 5: Pre-training corpus. Data mixtures used for pre-training, for each subset we list the sampling proportion, number of epochs performed on the subset when training on 260B tokens, and number of tokens.

sequently, our study does not incorporate any code-related subsets (e.g., GitHub and Stack Exchange).

In the previous work (Lee et al., 2022; Li et al., 2023), the importance of a high-quality pre-training corpus for model generalization performance is illustrated. Thus, we follow the instructions in Chen et al. (2023); Biderman et al. (2023) to deduplicate the corpus. As shown in Table 5, the corpus consists of eight main subsets: (1) **CCNews**. We preprocess and randomly sample a news subset of CommonCrawl[4], which contains news articles all over the world, ranging from 2016 to 2023. (2) **C4** (Raffel et al., 2020) is a publicly available corpus which contains massive web texts. We sample and retain the English documents from it. (3) **Wikipedia** is a widely used corpus for pre-training language models that contains high-quality knowledge. Therefore, we employ all English documents and increase its sampling probability during pre-training. (4) **Books** is an open-source subset in RedPajama, which we utilize the entire corpus. And follow the advice in Touvron et al. (2023), its sampling probability is also be raised. (5) **Law** is derived from the FreeLaw project included within the Pile. It is a publicly available resource for academic studies in the legal realm. (6) **ArXiv** is a collection of research papers across multiple fields (e.g., computer science, math). As ArXiv papers are written in LaTeX, which is a typesetting language, we parse these files and keep the text parts for pre-training. (7) **USPTO** contains background sections from patents granted by the United States Patent and Trademark Office, which incorporates a large number of technical documents. (8) **Other** corpus is composed of three subsets in Pile, i.e., Enron Emails, PhilPapers and NIH Grant Abstracts.

[4]https://data.commoncrawl.org/CC-NEWS/index.html

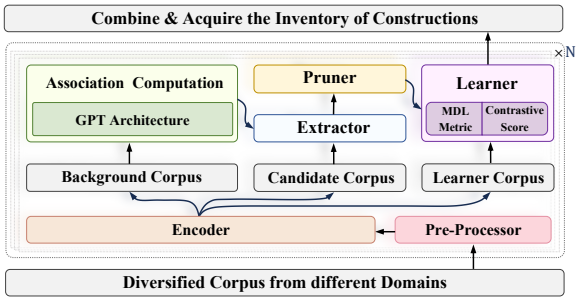

Figure 8: Overview of the *CxGLearner* framework.

D Implementation of CxGLearner

In Section 2, our induction framework[5] is introduced for automatically searching constructions from the corpora. We will elaborate on the specifics of the framework for better comprehension.

D.1 Overall of Induction Framework

As shown in Figure 8, our framework initially preprocesses the corpus across various domains. Then the processed corpus from each domain is segmented into three distinct parts for different stages. During the first stage, the background corpus serves to pre-train a language model, referred to as the Association Strength Estimator (ASE). The ASE assesses the association strength between sequences and slots, and it also helps generate representations for candidates. Subsequently, a recursive search with the ASE is conducted on the candidate corpus to extract potential candidates, which are then pruned in the hypothesis space. Finally, we apply heuristic algorithms to optimize potential candidates, simulating psychological principles, with the aim of deriving the optimal set of constructions for each domain. Ultimately, the generic construction inventory is determined by synthesizing the construction sets across all domains.

D.2 Pre-Processer and Encoder

Initially, we need to pre-process the corpus to avoid impacting the encoder. Thus, a Unicode checker is used to fix broken Unicode characters and the white spaces in corpus are also normalized for uniformity. Then we substitute the format-specific information in the text (e.g., email, phone) with special tags, such as <*EMAIL*> and <*PHONE*>.

We then partition the corpus into three distinct parts, which are in the ratio 6:3:1 for background, candidate and learner corpus, respectively. After

[5]Our code of induction framework will be publicly available at https://github.com/xlxwalex/CxGLearner

that, these sub-corpora are encoded into slots at different levels of abstraction. In this work, we encode the corpus with three levels, i.e., lexical, Universal Part-of-Speech tags (UPOS; Petrov et al., 2012) and Language-Specific Part-of-Speech tags (XPOS; Marcus et al., 1993). And the UPOS and XPOS provide coarse-grained as well as fine-grained slots for the syntactic level, respectively. Though the state-of-the-art models have reached the human ceiling on the part-of-speech tagging task and are close to solving the problem completely (Manning, 2011), we employ the voting method of ensemble learning (Zhou, 2012) to synthesize the results of multiple parsers (e.g., spaCy and Stanford Parser; Honnibal et al., 2020; Manning et al., 2014) to obtain results as robust as possible for the syntactic level slots. Since the granularity of slots at the syntactic level is at the word level, whereas the slots at the lexical level have sub-word granularity, we assign the same syntactic level slots to all sub-words of a specific word, and their intervals are tracked with the masks to align them at different levels.

D.3 Pre-training Procedure of ASE

Prior works (Dunn, 2017, 2019) have utilized an association-based approach to assess the constraints between adjacent slots, which only focus on localized features. A potential approach for addressing a global perspective is to utilize the entire generated potential construction sequence as context in determining whether the next slot satisfies the association condition instead of just focusing on the adjacent slots. However, this can introduce significant complexity due to a large solution space.

Thus, as described in Section 2.2, we propose a PLM-based Association Strength Estimator (ASE) to assess the slot constraints. The ASE is based on GPT-2 architecture (Radford et al., 2019), which is pre-trained on background corpus for each domain via the Next Slot Prediction (NSLP) task. During pre-training, we select slots randomly from different levels for the encoded pre-trained corpus. In order to accommodate to slots of different granularity, we adapt the Whole Word Masking (WWM; Cui et al., 2021) technique. If a slot at the syntactic level is selected and its corresponding lexical slot is a sub-word token identified by the mask, then the entire word is selected as a whole. The ASE is pre-trained for multiple epochs on the corpus, and different slots are selected dynamically at each epoch so that more combinations of slots can be captured. Then we feed the sequence into the ASE for inferencing, whose output can be exploited as an estimate of the association strength between the sequence and an arbitrary slot. Furthermore, ASE is also employed to generate representations of constructions for the learner component.

D.4 Candidate Extraction and Pruning

The component of extractor is employed to generate potential candidates from the candidate corpus. As introduced in Section 2.3, all the slots in each sentence are utilized as starting positions in the potential sequence, and then a recursive search is utilized to add slots to the sequence that satisfy the association condition (the ASE and the strategy of nucleus set are applied to determine the addition of slots, which is elaborated in Section 2.3). In search procedure, we use a breadth-first traversal to determine whether the constraints hold among the current potential sequence and all the hierarchical slots in the next position. Meanwhile, as long as the condition of adding slots is satisfied, the added sequence will be included to the result set as a potential construction sequence.

Under this setting, it might result in some of the candidates with non-optimal boundaries. Thus, we implement a pruning procedure to refine the hypothesis space of these candidates. The process comprises four stages of pruning: (1) Rule pruning. It filters out constructions that fail to comply with the set rules, such as cross-sentence. (2) Frequency pruning. In usage-based perspective, constructions are the patterns that occur with sufficient frequency. Thus, this pruning stage aims at preserving as many generic candidates as possible. (3) Horizontal pruning. It tends to keep longer potential candidates. More specifically, if a short potential sequence is only contained within a longer candidate sequence, then we just keep the longer one. (4) Vertical pruning. It is employed to eliminate candidates that include superfluous abstraction slots. If a slot in a construction can only be instantiated to one possible word and the instantiated construction is included in the result set, we only retain the instantiated construction.

D.5 Optimization of Construction Inventory

After extracting potential construction candidates, it is necessary to further refine the candidate set in learner component, as not all usage-based constructions are worth storing (Jackendoff, 2003). There are four psychological principles considered essential for the organization of constructions in a

language. These principles can be integrated into our efforts to optimize the potential construction set (Goldberg, 2003; Goldberg et al., 2005). We separate the four principles into two groups of optimization objectives as follows: (1) The principles of **Maximized Expressive Power** and **Maximized Economy**. From the usage-based perspective (Dunn, 2019; Goldsmith, 2006), these two principles are interdependent, striking a balance between construction complexity (the encoding size of construction list) and descriptive adequacy (the encoding size of a corpus given the grammar). Thus, the minimum description length (MDL; Rissanen, 1978) is aptly suited for modeling these principles. The MDL metric can be defined as:

$$S_{\text{MDL}} = \gamma S_{\text{Com}}(G) - \eta S_{\text{Des}}(T|G) \quad (2)$$

where G denotes the list of constructions, while T refers to the learner corpus. The γ and η serve to balance the two terms. The construction complexity, defined as the total information content within the construction list, is represented by the first term:

$$S_{\text{Com}}(G) = -\sum_{g \in G} \sum_{s=1}^{|g|} \frac{1}{|g|} \log \frac{1}{N_s} \quad (3)$$

Here, g denotes a construction comprising $|g|$ slots. N_s indicates the number of instantiations for each slot: it is 1 for a lexical slot, and for the abstract slot, it refers to the number of tokens that can fill it. The second term is intended to assess descriptive adequacy and is defined as follows:

$$S_{\text{Des}}(T|G) = \frac{1}{n} \sum_{j=1}^{n} \log \mathcal{R}_j^{cov} - \log \mathcal{R}_j^{ove} \quad (4)$$

where n denotes the number of sentences in the corpus. The constructions within each sentence are scanned, with $\mathcal{R}_j^{cov}$ representing the coverage while $\mathcal{R}_j^{ove}$ indicating the overlap of matched constructions in the j-th sentence.

(2) The principles of **Maximized Motivation** and **No Synonymy**. These principles also symbolize a pair of mutually dependent goals, indicating that if two constructions are syntactically related, they also bear a semantic relationship to a certain degree, and the converse applies as well. Thus, in a similar vein to contrastive loss (Chen et al., 2020), we propose the concept of contrastive scoring:

$$S_{\text{C}} = -\sum_{g \in G} \log \frac{\sum_{g^+} \exp\left(\sigma_{g,g^+} \textsc{Sim}(h_g, h_{g^+})/\tau\right)}{\sum_{g^-} \exp\left(\textsc{Sim}(h_g, h_{g^-})/\tau\right)} \quad (5)$$

Benchmark	Type	# Num.	Answer
Arc-E	QA	2,376	MC
Arc-C	QA	1,176	MC
OBQA	QA	2,000	MC
LogiQA	QA	2,604	MC
PIQA	QA	3,084	MC
HellaSwag	Reasoning	10,003	MC
BoolQ	RC	3,245	Boolean
WSC	CR	146	MC
QNLI	NLI	10,926	MC
LAMBADA	Generation	5,153	LL

Table 6: Statistics for benchmark tasks. For task types, QA denotes question answering task, NLI indicates natural language inference task, RC stands for reading comprehension task, while CR refers to coreference resolution task. In answer type, MC stands for multi-choice, while LL refers to loglikelihood.

syntactic distance σ is determined by the proportional overlap length between two constructions. For abstract slots, overlap includes the inclusion relation, but this is counted as half the length of the overlap. If the syntactic distance exceeds 0, g^+ is a positive sample of g, otherwise negative g^-. ASE is employed to provide the representations h of the constructions, while SIM is utilized to compute the semantic similarity (e.g., cosine similarity). τ is a temperature coefficient for adjusting the sensitivity to differences between positive and negative pairs.

We employ heuristic algorithms (e.g., Simulated Annealing; Kirkpatrick et al., 1983) to optimize both objective $\min\{\alpha S_{\text{MDL}} + \beta S_{\text{C}}\}$, where α and γ are weighting factors. Then the optimal construction inventory for each domain is obtained.

D.6 Generic Construction Synthesization

Following the previous three stages, we obtain a proprietary set of constructions for each domain. In order to acquire the generic inventory, we synthesize the set of constructions across all domains. More specifically, we retain only those constructions that emerge in at least two domains. The fact that these constructions exist across domains is evidence of their general applicability. After that, the generic inventory of constructions is available.

E Details of Benchmark Tasks

In Section 4, we conduct experiments for comparison on various benchmark tasks, including 10 tasks from Language Model Evaluation Harness (Gao et al., 2023) framework. As shown in Table 6, we present the statistics for these benchmarks.

Configuration Key	Value
num-layers	24
hidden-size	1024
intermediate-size	4096
num-attention-heads	16
num-head-size	64
num-query-group	4
attention-dropout	0.0
hidden-dropout	0.0
pos-emb	rotary
rotary-percentage	1.0
rotary-condense-ratio	4
sequence-length	2048
eval-interval-steps	2000
trainig-strategy	DDP
global-batch-size	512
accumulation-steps	32
gradient-clipping	1.0
init-method	GPT-NeoX (Black et al., 2022)
optimizer.type	AdamW
optimizer.params.lr	4e-4
optimizer.eps	1e-8
optimizer.params.betas	[0.9, 0.95]
lr-decay-style	cosine
min-lr	$0.1 * $ optimizer.params.lr
vocab-size	63595
warmup-steps	2000
flash-attn	True
norm	RMSNorm
norm-eps	1e-5

Table 7: The implementation details for pre-training.

Model	Params	E-Params
GPT-2	355M	51M
OPT	331M	26M
Pythia	354M	52M
BLOOM	**560M**	**257M**
CoELM	533M	130M

Table 8: Computational complexity analysis of models. **Params**, **E-Params** and **MACs** represent the number of parameters, the number of parameters for embedding layer and multiply–accumulate operations, respectively.

Physical Interaction Question Answering (PIQA; Bisk et al., 2020) task primarily focuses on evaluating understanding of physical common sense. The Boolean Questions (BoolQ; Clark et al., 2019), Question-answering NLI (QNLI) and Winograd Schema Challenge (WSC; Levesque et al., 2012) tasks are all included in GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019). BoolQ is a question answering task that involves analyzing a brief passage from Wikipedia followed by a yes/no question pertaining to the text, while WSC presents a coreference resolution task where each example comprises a sentence that includes a pronoun and a list of noun phrases extracted from that sentence. Then the PLMs are requested to accurately identify the noun phrase that the pronoun refers to from the provided options. And QNLI (White et al., 2017; Demszky et al., 2018) is converted from question-answering datasets (e.g., SQuAD). The task is to determine whether the context sentence contains the answer to the question. HellaSwag (Zellers et al., 2019) is designed to assess the ability of the models to understand both grounded and temporal aspects of common sense. The AI2 Reasoning Challenge (ARC; Clark et al., 2018) contains two subsets, i.e., ARC-Easy (Arc-E) and ARC-Challenge (Arc-C), which are centered on science exam questions that necessitate complex reasoning under different levels of difficulty. OpenBookQA (OBQA; Mihaylov et al., 2018) is a question-answering dataset modeled after open book exams for assessing human understanding of a subject. LogiQA (Liu et al., 2020) is a task for testing human logical reasoning, covering multiple types of deductive reasoning. LAMBADA (Paperno et al., 2016) serves as a benchmark dataset for assessing the proficiency of language models in comprehending text through a word prediction challenge.

F Hyper-parameters and Settings

In Table 7, we provide the full configuration details for pre-training our CoELM. We implement our CoELM using PYTORCH and LIT-GPT framework [6]. Then CoELM is pre-trained with 250,000 steps from scratch on a 8 $\times$ NVIDIA GeForce RTX 4090 device with about 3,000 GPU hours.

As shown in Table 8, we compare the number of parameters of our CoELM with other baseline models. Though we utilize hyper-parameters that are harmonized with the other language models, it can be observed from the results that our CoELM has more parameters than GPT-2, OPT, and Pythia, but fewer than BLOOM. This is mainly due to two aspects: (1) the size of our embedding layer consists of two parts, the tokens in vocabulary and the inventory of constructions. Therefore, the embedding layer has a larger number of parameters. (2) Our CoELM exploit the Llama architecture, in which SwiGLU is used. Thus, each layer will have 4M more parameters than the baseline models, bringing about a gap in the number of parameters.

[6] https://github.com/Lightning-AI/lit-gpt

Domain	Corpus	# Candidate	# Inventory
Commercial	Mail	20,829	8,036
Knowledge	Wiki	80,331	13,668
Academic	ArXiv	69,262	12,001
Reportorial	CCNews	61,583	10,309
Professional	Law	85,147	15,014
	USPTO	77,819	13,161

Table 9: Statistics for constructions. **# Candidate** denotes the number of construction candidates, while **# Inventory** is the number of constructions in inventory.

G Statistics of Construction Inventory

As described in Section D, the inventory of constructions is synthesized across various domains for acquiring the generic constructions. In order to be applicable to the pre-training process, we identify five primary domains for domain-specific construction acquisition, which are shown in Table 9. For each domain, we sample the corpus of 3B tokens, except for the commercial domain, where we exploit the entire corpus since the Mail corpus is relatively small. The candidate constructions generated by the extractor component and the inventory of pruned and optimized constructions are then reported. After that, we synthesize constructions across these domains and obtain 13,262 generic constructions in the inventory.

As shown in Table 10, we present some representative constructions for the five domains. Semi-idiomatic and idiomatic constructions are used for demonstration as they could be observed more intuitively. For example, "mail" is widely used for contacting in the commercial domain. And the construction "quote–of–the–NOUN" is often exploited to make references to correspondence and matters, while "to–contact–PROPN" is utilized to indicate who can be contacted. Meanwhile, "reservation–of—rights" and "this–section–AUX–not–prior" are typical constructions from the legal and professional domain. These observations illustrate the validity of our framework.

H Detailed Configuration of ASE

In our induction framework, we utilize ASE to simulate unidirectional association strength (Dunn, 2019) for assessing slot constraints, which is introduced in Section 2.2 and Appendix D. We employ GPT-2 architecture to implement ASE and pre-train it with a next slot prediction task. To explore the correlation between the output of ASE and the association strength, we pre-train ASE on academic domain with ArXiv corpus (Table 9) and conduct comparative experiments for different sizes of ASE.

Domain	Example
Commercial	quote–of–the–NOUN to–contact–PROPN please–VERB–DT–NOUN
Knowledge	AUX–serve–as sponsored–by–the–NOUN NN–directed–by–PROPN
Academic	estimated–from–DET–NOUN analytic–expression–IN uniquely–determined–by
Reportorial	commissioner–for–NOUN after–allegedly–VBG holdings–in–shares–of–NOUN
Professional	reservation–of–rights claimed–IN–NOUN–AUX discharge–of–NOUN this–section–AUX–not–prior

Table 10: Examples of constructions across five different domains.

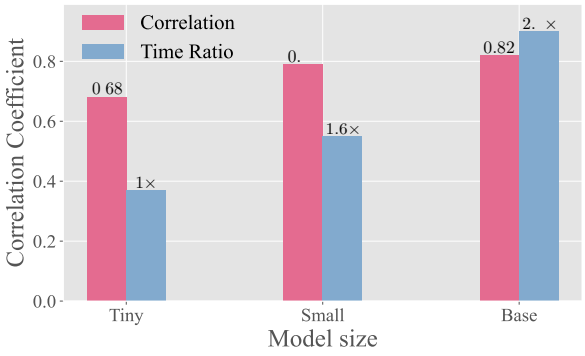

Figure 9: Comparison among different sizes of ASE.

We compare three different sizes of the ASE and keep the other hyper-parameters the same, which are denoted as Tiny, Small, and Base, respectively. For the hyper-parameters, we set hidden size to 768 with 12 heads, while the intermediate size is set to 3072. Meanwhile, the numbers of layers of these three sized ASEs are set to 4, 6 and 12, respectively. Then we randomly select 10,000 pairs of slots in the encoded candidate corpus of the Arxiv for comparative experiment.

As shown in Figure 9, we demonstrate the correlation coefficients between the outputs of ASE and the association strengths, as well as the inference times for different size of ASEs. The correlation coefficients of Small and Base are relatively close to each other, both indicating a high degree of cor-

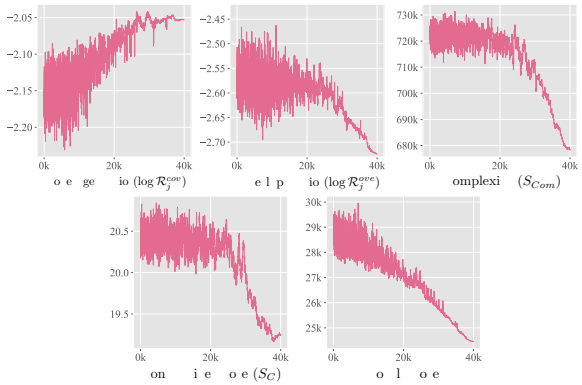

Figure 10: The curve of objectives in learner module.

relation. The correlation coefficient for Tiny is significantly lower than Small and Base, which can be considered moderately correlated. However, larger size models also require longer inference times as well as more computational resource consumption. We can observe that the inference time of Base is 2.4 times that of Tiny. For the trade-off between high correlation and inference speed, the Small ASE with 6 layers is employed to assess the slot constraints. Meanwhile, the high correlation between the output of ASE and association strength can also demonstrate the feasibility of our ASE.

I Optimization Process of Learner

As we described in Section D.5, the learner component is employed to optimize the potential construction candidates. Thus, we utilize MDL metric and contrastive score that correspond to two groups of psychological principles for optimization.

We illustrate the trend of each objective during the optimization process for academic domain in Figure 10. The first row of the figure presents the components of the MDL metric, which we introduced in Equation 2 ~ 4. In Figure 10, we can observe that as the optimization proceeds, the coverage ratio of the constructions over the learner corpus keeps increasing while the overlap between constructions decreases, which is aligned with our optimization objectives. Meanwhile, the construction complexity and contrastive score are also descending. Since the scales of the objectives are different, we introduce balancing coefficients to allow them to be close to each other in order of magnitude. Ultimately, the total objective score converges in the optimization procedure, demonstrating that our learner component can achieve the optimization goals for all psychological principles.

Model	Perplexity	Accuracy
Pythia-1B	32215.6	0.00
TinyLlama	320.9	0.00
CoELM	**18.3**	**0.36**

Table 11: The few-shot experimental results of wug test via the prompt of Language Model Evaluation Harness framework. The lowest perplexity and highest accuracy is in **bold**.

J Few-shot Setting for Wug Test

In Wug Test (Section 4.3), for language models other than CoELM, we do not directly utilize the prompt from Language Model Evaluation when conducting few-shot setting. Instead, we exploit the prompts suggested by Weissweiler et al. (2023b). This is due to the discovery that utilizing other unseen samples from the Wug Test as prompts would lead to significantly diminished performance of the baselines. As shown in Table 11, both Pythia-1B and TinyLlama exhibit high perplexity and low accuracy in such a few-shot setting. In contrast, our CoELM significantly outperforms them. We believe this is due to the constructional information introduced during the pre-training, which can guide the model to recognize the part of speech of unseen words and to perform tense transformation. This illustrates that with the aid of constructional information, the language models may achieve greater robustness towards unseen words.