LM-LEXICON: Improving Definition Modeling via Harmonizing Semantic Experts

Yang Liu[1*†] **Jiaye Yang**[2*] **Weikang Li**[3]
Jiahui Liang[2] **Yang Li**[2] **Lingyong Yan**[2]
[1]BIGAI [2]Baidu Inc. [3]Peking University
liuyang@bigai.ai

https://lm-lexicon.github.io
https://huggingface.co/LM-Lexicon

Abstract

We introduce LM-LEXICON, an innovative definition modeling approach that incorporates data clustering, semantic expert learning, and model merging using a sparse mixture-of-experts architecture. By decomposing the definition modeling task into specialized semantic domains, where small language models are trained as domain experts, LM-LEXICON achieves substantial improvements (+7% BLEU score compared with the prior state-of-the-art model) over existing methods on five widely used benchmarks. Empirically, we demonstrate that 1) the clustering strategy enables fine-grained expert specialization with nearly 10% improvement in definition quality; 2) the semantic-aware domain-level routing mechanism achieves higher expert efficacy (+1%) than conventional token-level routing; and 3) further performance gains can be obtained through test-time compute and semantic expert scaling. Our work advances definition modeling while providing insights into the development of efficient language models for semantic-intensive applications.

1 Introduction

Defining terms (Fig. 1) is the first step toward building a lexicon for a language (Pustejovsky and Boguraev, 1993). Precise definitions should be formed as summarized and human-readable sentences that capture the main sense of a term. Modern language use demands continuous updates to include new terms, novel senses, meaning shifts, and domain knowledge (Hogeweg and Vicente, 2020), yet traditional lexicon construction remains labor-intensive (Ahlswede, 1985). To address this challenge, definition modeling (DM) has emerged as a promising approach, where definitions are automatically generated based on the target term and its context (Giulianelli et al., 2023, *inter alia*).

Figure 1: Four examples of the **term**, context (input), and definition (output) for definition modeling task.

While existing DM approaches yield reasonable results, they face several key limitations. First, current methods struggle to capture subtle and rare word senses, resulting in incomplete semantic coverage (Huang et al., 2021; Giulianelli et al., 2023; Periti et al., 2024). Second, even frontier large language models (LLMs), despite their strong language understanding capabilities, tend to generate definitions that are either overly generic or excessively specific (Jhirad et al., 2023; Yin and Skiena, 2023; Almeman et al., 2024). Third, existing methods often fail to handle terms that exhibit different meanings across domains (*e.g.*, technical vs. general usage), a phenomenon known as *semantic heterogeneity* (Huang et al., 2021). Recent attempts to address this limitation through domain adaptation (Zhang et al., 2022) or multi-task learning

*Equal contribution.
†Correspondence to: liuyang@bigai.ai

(Kong et al., 2022) have shown limited success. These challenges point to a fundamental limitation in current dense language models: their architecture forces much semantic representation to share the same neurons (*i.e.*, superposition) (Elhage et al., 2022), making it difficult to maintain precise, domain-specific meaning representations (Bricken et al., 2023). This architectural constraint affects their ability to generate accurate definitions when words have distinct meanings across different domains.

To mitigate these issues, we propose LM-Lexicon (**L**anguage **M**odel as **Lexicon**), which learns to perform DM covering multiple domains, adapting diverse definition genres with a scalable mixture-of-experts (MoE) architecture. Unlike prior work, such as BTX (Sukhbaatar et al., 2024) and LLaMA-MoE (Zhu et al., 2024), **our method incorporates data clustering, semantic expert-specialized MoE, and domain-level sequence routing, obtaining significant performance gains in DM benchmarks.** As depicted in Figure 2, instead of training directly on raw definition corpora, our method trains multiple semantic experts parallely, merges them by composing their specialized weights, and routes test samples with the introduced semantic-aware router for inference.

Our contributions can be summarized as follows:

- We propose LM-Lexicon, a framework for definition modeling by harmonizing inherent heterogeneity in lexical semantics. It allows specialized semantic experts to be integrated for domain updates, enabling generalization to new domains, or collapsing back to a single expert for efficient inference.

- We design a domain-level sequence routing policy in LM-Lexicon. This method routes representation of samples informed by fine-grained information via semantic domains identified with pre-hoc auto clustering.

- Extensive experiments across five benchmarks validate the effectiveness of LM-Lexicon. Notably, in automatic evaluation, LM-Lexicon shows up to 10% improvement over strong baselines. Furthermore, LM-Lexicon excels across most criteria in human evaluation, particularly outperforming frontier LLMs in semantic-intensive scenarios, where even many-shot setups fail to produce appropriate definitions.

2 Related Work

Upcycling to Mixture-of-Experts. On the aspect of model efficiency and expressiveness, Fedus et al. (2022); Jiang et al. (2024); Shao et al. (2024) focus on designing efficient MoE architecture with token-level router. From the expert specialization aspect, Li et al. (2022) introduced Branch-Train-Merge (BTM) that learns expert LMs specialized to different domains and Sukhbaatar et al. (2024) developed Branch-Train-MiX (BTX), which composes a set of specialized LMs by their feed-forward networks. In addition, Zoph et al. (2022); Jiang et al. (2024); Petridis et al. (2024) revealed the efficacy of expert specialization at the lexicon, structured syntactic, and semantic domain level, respectively. However, these works adopt conventional routing schemes, such as TopK routing, rather than exploring those better suited for semantic-intensive tasks.

Definition Modeling. Several early studies on DM (Noraset et al., 2017; Ni and Wang, 2017; Gadetsky et al., 2018; Ishiwatari et al., 2019, *inter alia*) leveraged pre-trained word embeddings as global or local contexts of a term, to generate definitions of the given target word. Then Huang et al. (2021); Kong et al. (2022); Zhang et al. (2022); Giulianelli et al. (2023); Periti et al. (2024) propose methods for DM using Transformer-based Seq2Seq LMs (*e.g.*, T5) and Causal LMs. In the era of LLM, Jhirad et al. (2023) and Yin and Skiena (2023) used large language models such as GPT-3.5 and GPT-4 to perform DM with in-context learning tailored to diverse domains. Periti et al. (2024) explored training causal LMs to generate with instruction tuning; however, they still lack a detailed quality evaluation and comphrehensive comparison with baselines.

3 Methodology

In this section, we present the details of our proposed LM-Lexicon framework. §3.1 introduces the formulation to illustrate the main idea. In §3.2, we illustrate the design of semantic expert specialization, followed by model merging in §3.3.

3.1 Overview of LM-Lexicon

Given a seed model $\mathcal{M}$ that has been pre-trained, our goal is to improve its multi-domain performance in lexical semantics. As shown in Fig. 2, the framework of LM-Lexicon consists of two components: **(1) semantic expert special-**

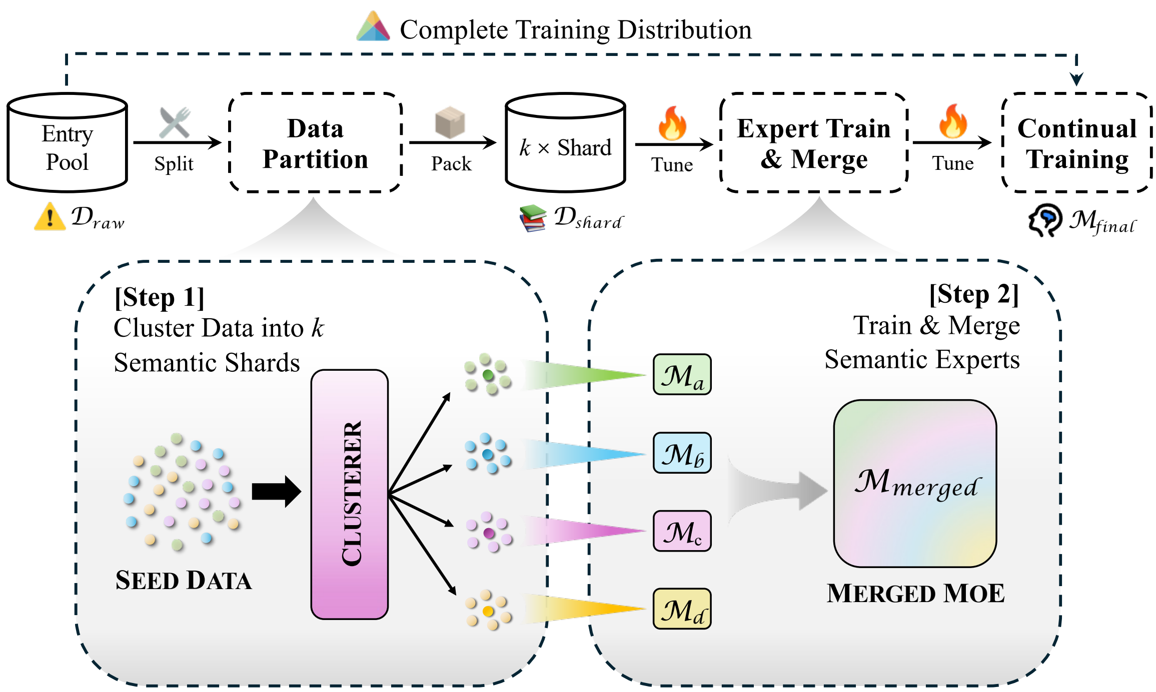

Figure 2: Diagram of LM-Lexicon (*i.e.*, *Split-then-Merge*) pipeline.

ization and **(2) MoE model merging**. The proposed method contains three stages, training data partitioning, parallel expert training, and separate experts merging, i.e., a *Split-then-Merge* pipeline. Considering the heterogeneity of glosses, we split the training data into semantically distinctive clusters to facilitate expert learning. To model various domains, we use separate models to learn domain-specific knowledge asynchronously. To perform the DM task generally, we merge these experts into a single MoE model for further fine-tuning.

3.2 Learning Domain-specific Semantic Experts

Dataset Construction. Training data $\mathcal{D}$ consists of triplets $\langle c, t, d\rangle$, where c represents the context in which the term is used (either a sentence or phrase), t denotes the term itself, and d is its reference definition. A concatenated sequence is then formatted using the prompt template $p(\cdot, \cdot)$ as input. Specifically, we follow Giulianelli et al. (2023) to use $p :=$ <BOS>"$\{\{c\}\}$" WHAT IS THE DEFINITION OF "$\{\{t\}\}$"<EOS> as the prompt template.

Clustering. LM-Lexicon begins with the training data partitioning since merging without it could lead to a group of homogeneous experts. To cluster training data, we calculate the embeddings of $p(c, t)$ in each training sample with *nvidia-embed-v2* (Lee et al., 2025), and then cluster with balanced k-means (Malinen and Fränti, 2014). This process results in N clusters in terms of lexical semantics, each related to a semantic domain such as adjectives and proper nouns (see Fig. 3), corresponding to partitioned training datasets $\mathcal{D} := \{\mathcal{D}_1, \ldots, \mathcal{D}_N\}$. It also produces N cluster centroids $\{v_1, v_2, \ldots, v_n\}$. In the present study, we perform pre-experiments to determine the number of clusters and select $N = 4$ as the best cluster numbers by the cluster cohesion and separation in the DM scenario (See Appendix §C.1), as well as considering the training and inference efficiency.

Experts Training. Initializing from a seed model $\mathcal{M}$, we train $N \times$ LMs: $\{\mathcal{M}_1, \ldots, \mathcal{M}_N\}$ as experts, with each model $\mathcal{M}_i$ being trained on the corresponding dataset $\mathcal{D}_i$, using the negative log-likelihood (NLL) loss in Eq. 1:

$$\mathcal{L}_{\text{NLL}} = -\,\mathbb{E}_{(c,t,d)\sim\mathcal{D}}\left[\log \mathcal{P}(\hat{d} \mid p(c,t))\right]. \quad (1)$$

Here, $\hat{d}$ denotes the definition predicted by the model, given the prompt $p(\cdot, \cdot)$. We employ a loss-masking strategy to omit the tokens of prompt during loss computation, ensuring that gradients are only propagated through tokens in the part of predicted definition. When expert training finished, we end up with N different LMs, with each specialized in a domain $\mathcal{D}_i$.

3.3 Merging Experts into a Unified MoE

After all domain experts are obtained, previous works either average the final output distributions of experts to generate next token (Gururangan et al., 2023) or select experts by determining which domain the input belongs to at the test time (Li et al., 2022). Differently, we perform MoE Upcycling by merging the weights of experts, aiming at mixing model capabilities across diverse domains.

Model Merging. We combine semantic experts into a unified MoE to exploit the parametric domain capability (Sukhbaatar et al., 2024; Zhou et al., 2025). In the composition, LM-LEXICON brings together the feed-forward networks (FFNs) of the expert models as expert layers in MoE and averages the remaining parameters. Specifically, if $\text{FFN}_i^\ell(x)$ is the FFNs at the ℓ-th layer of the i-th expert $\mathcal{M}_i$, then the combined MoE layer for input representation x at layer ℓ will be computed as:

$$\text{FFN}_{\text{MoE}}^\ell(x) = \sum_{i=1}^{N} \mathcal{G}(x) \cdot \text{FFN}_i^\ell(x). \tag{2}$$

where $\mathcal{G}(\cdot)$ is a semantic domain-level router. During both training and inference, the input representation x will be routed to the nearest centroid by computing its pairwise cosine similarity with each semantic label (*i.e.*, the centroid of a domain cluster), as illustrated in §3.2. $\mathcal{G}(\cdot)$ usually has a sparse output and hence switches on only some experts. In LM-LEXICON, we start from top-k (k = 2) routing (Shazeer et al., 2017), where $\mathcal{G}(x) = \text{Softmax}(\text{TopK}(W^\ell x))$, where W^ℓ is a linear transformation in router. For multihead self-attention (MHA) sublayers and the remaining parameters (*e.g.*, embedding layer), we average the weights of domains. The merging process of MoE model is provided in Algorithm 1.

The above merging model into a MoE introduces router $\mathcal{G}$ with new parameters W^ℓ, which requires further learning to make optimal choices. To enhance semantic-aware experts after merging, we continue to slightly fine-tune the router $\mathcal{G}$ and expert layers to coordinate them in the semantic representation space.

Algorithm 1 Compose MHA and MLP modules for each decoder layer ℓ in LM-LEXICON.

Input: Domain Experts $\mathcal{E} := \{e_1, e_2, \ldots, e_n\}$.
Output: LM-LEXICON-MOE ($\mathcal{M}$)

procedure MODULES-COMPOSER($\mathcal{E}$)
  $\mathcal{M} \leftarrow \emptyset$ ▷ INIT STATE DICT
  for $e_i \in \mathcal{E}$ **do** ▷ ITERATE EACH EXPERT
    $i \leftarrow \text{GetExpertIdx}(e_i)$
    /* Retrieve MHA and MLP weights */
    $\theta_{mha}, \theta_{mlp} \leftarrow \text{HookWeights}(e_i)$
    for $\theta \in \{\theta_{mha}, \theta_{mlp}\}$ **do**
      if IsRouterLayer(θ) **then**
        /* Get formatted layer name */
        $n \leftarrow \text{FormatName}(\theta, i)$
        $\mathcal{M}[n] \leftarrow \theta$
      else ▷ AVERAGE θ OF MODULE
        $\mathcal{M}[n] \leftarrow \mathcal{M}.\text{get}(n, \mathbf{0}) + \theta/|\mathcal{E}|$
  return $\mathcal{M}$

4 Experiments

4.1 Implementation Details

Datasets. We use the benchmarks introduced in Ishiwatari et al. (2019)(see Table 1), which consist of four small datasets and 3D-EX from Almeman et al. (2023) (see details in §A).

- **WordNet** (Noraset et al., 2017) is an online dataset[1] of terms, definitions, and examples.
- **Oxford** (Gadetsky et al., 2018) is built on the widely used online oxford dictionary[2].
- **Wikipedia**[3] (Ishiwatari et al., 2019) is introduced to test the model capacity on the description of phrases, rather than words.
- **Urban** (Ni and Wang, 2017)[4] contains terms of internet slang and urban words.
- **3D-EX** (Almeman et al., 2023) is the largest English definition modeling dataset[5] which comprises many well-known DM resources, including the four mentioned datasets.

Note that we perform clustering only on 3D-EX and use the resulting four clusters for finetuning and merging semantic experts.

Compared Baselines. Llama-3-8B (Dubey et al., 2024) is used as the seed model for asynchronous expert training. We select three types of strong baseline methods for comparison purposes.

[1] https://wordnet.princeton.edu
[2] https://en.oxforddictionaries.com
[3] https://www.wikidata.org
[4] https://www.urbandictionary.com
[5] https://github.com/F-Almeman/3D-EX

	WordNet	Oxford	Wikipedia	Urban	3D-EX
genre	formal	formal	web	idiom	misc.
domain	synset	lexicon	encyclopedia	slang	multi
publish year	2017	2018	2018	2017	2023
# $\mathcal{S}^t_{\text{train}}$	$13,883$	$97,855$	$887,455$	$411,384$	$1,309,312$
# $\mathcal{S}^t_{\text{valid}}$	$1,752$	$12,232$	$44,003$	$57,883$	$513,789$
# $\mathcal{S}^t_{\text{test}}$	$1,775$	$12,232$	$57,232$	$36,450$	$450,078$
# glo. per term	1.75 ± 1.19	2.99 ± 4.41	5.86 ± 78.25	2.11 ± 2.92	6.00 ± 53.78
# tok. per term	1.00 ± 0.00	1.00 ± 0.00	1.85 ± 0.93	1.44 ± 0.72	1.45 ± 0.78
# tok. per ctx.	5.79 ± 3.44	19.02 ± 9.18	19.68 ± 6.31	11.36 ± 6.02	18.82 ± 9.99
# tok. per glo.	6.64 ± 3.78	11.41 ± 7.13	5.97 ± 4.51	11.02 ± 6.86	8.97 ± 6.76
% overlap rate	0.00 / 0.00	80.72 / 0.09	0.00 / 0.00	20.62 / 20.56	0.00 / 0.00

Table 1: For datasets used in this paper, we report the mean and standard deviation of per-term, per-context, and per-gloss statistics. We report the number of terms of samples denoted $\mathcal{S}^t_*$ for train, valid, and test splits in each dataset. The lexical overlap of each dataset is computed with $|\mathcal{S}^t_{\text{train}} \cap \mathcal{S}^t_{\text{test}}| \;/\; |\mathcal{S}^t_{\text{test}}|$. Specifically, the % is computed by intersection rate of term occurrence and the % is computed by intersection rate of pair-wise "term $\oplus$ gloss".

- **Supervised Seq2seq LM:** We reproduce Rerank-T5 (Huang et al., 2021), Contrast-T5 (Zhang et al., 2022), SimpDefiner (Kong et al., 2022), MDM-T5 (Zhang et al., 2023), and Flan-T5-Def (Giulianelli et al., 2023).

- **Supervised Causal LM:** We report the in-distribution results of LlamaDictionary (Periti et al., 2024), which is finetuned on ***Llama-3-8B-Instruct***, and assess its out-of-distribution performance for the unseen domains.

- **Frontier Causal LM:** We test GPT-4-Turbo (Achiam et al., 2023), Gemini-1.5-Pro (Reid et al., 2024), and Claude-3-Opus (Anthropic, 2024) with random exemplar selection (Random-ICL) and retrieval-based exemplar ranking (Retrieval-ICL) based on Wu et al. (2023) in many-shot settings.

Training and Evaluation Details. We run instruction tuning on four clusters obtained from 3D-EX respectively. The models trained on four clusters of 3D-EX are merged through §3.3. After merging, we proceed to fine-tune the MoE model to learn routers using the full 3D-EX dataset. In addition, we perform instruction tuning on the four real-world datasets. The hyperparameters can be found in the Tab. 12. We run three times with seeds to report the mean results and the standard deviation, with seed $s_i \in \{21, 42, 84\}$. All experiments are conducted on 8 $\times$ NVIDIA H100. Model sizes and training FLOPs are reported in Table 6.

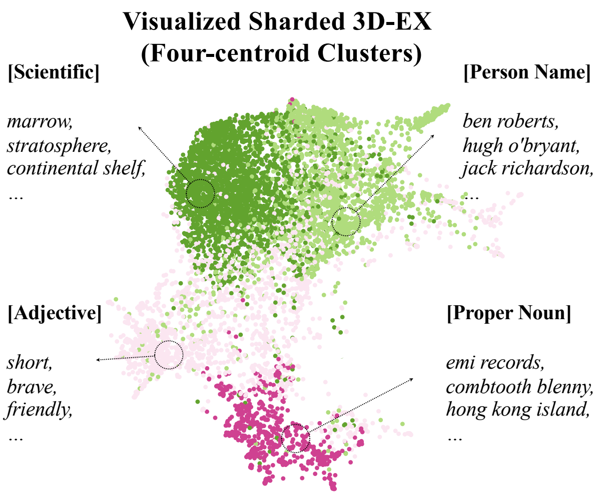

Figure 3: Four-cluster UMAP plot of 10K random definitions of terms in 3D-EX (§4). Each cluster is assigned manually with a **[label]** by their major constituents.

We employ metrics including (1) lexical n-gram-based: BLEU (Papineni et al., 2002), ROUGE-L (Lin, 2004), and METEOR (Lavie and Agarwal, 2007); (2) semantic-based: BERTSCORE (Zhang et al., 2019), MOVERSCORE (Zhao et al., 2019), and MAUVE (Pillutla et al., 2021). We reuse the implementation of BLEU in Huang et al. (2021), ROUGE and BERTSCORE used in Giulianelli et al. (2023), as well as the rest of metrics for evaluation. To further evaluate the effectiveness of our method, we perform a human evaluation described in §4.2.

4.2 Main Results

[6] We develop ad-hoc heuristic parser for proprietary models

	WordNet		Oxford		Wiki		Urban		3D-EX		Avg. Results
	BLEU	ROUGE	BLEU	ROUGE	BLEU	ROUGE	BLEU	ROUGE	BLEU	ROUGE	
Rerank-T5 (2021)♣	30.91	30.99	25.56	28.00	55.61	57.25	17.77	18.25	34.43	38.57	32.85 / 34.61
Contrast-T5 (2022)♣	30.81	26.27	22.51	28.18	55.26	42.27	17.53	16.34	34.27	37.62	32.07 / 30.13
SimpDefiner (2022)♣	28.91	20.47	23.48	29.59	44.03	49.26	13.54	15.37	32.08	31.57	28.40 / 29.25
MDM-T5 (2023)♣	31.18	32.55	24.16	27.68	54.33	55.83	17.53	17.18	32.67	32.38	31.97 / 33.12
Flan-T5-Def (2023)♣	31.96	40.45	21.34	32.39	13.82	23.97	5.33	10.61	26.43	25.12	19.77 / 26.50
LlamaDict (2024)♣	33.86	**43.50**	22.77	$\underline{36.46}$	14.38	25.29	15.70	14.51	24.56	26.11	22.50 / 29.17
GPT-4-TURBO											
↪ + *Random-ICL*	30.95	32.61	21.93	30.82	31.63	45.89	11.08	12.19	25.93	34.48	24.30 / 31.19
↪ + *Retrieval-ICL*	27.46	29.74	20.44	34.35	35.40	40.68	22.53	26.53	29.73	37.66	27.11 / 33.79
CLAUDE-3-OPUS											
↪ + *Random-ICL*	28.63	27.84	19.99	34.21	23.30	35.22	1.59	3.08	18.57	28.49	18.41 / 25.76
↪ + *Retrieval-ICL*	18.57	21.76	15.51	25.99	14.59	15.83	5.93	7.19	17.46	24.67	14.41 / 19.08
GEMINI-1.5-PRO											
↪ + *Random-ICL*	23.42	26.27	25.51	35.97	36.87	48.13	8.44	9.59	29.4	38.02	24.72 / 31.59
↪ + *Retrieval-ICL*	25.24	27.88	**28.10**	**36.98**	35.59	43.71	8.85	9.18	32.99	39.14	26.15 / 31.37
LM-LEXICON-DENSE (8B)											
↪ + *Zero-shot*	$36.99^{*}_{0.59}$	$37.83^{*}_{0.45}$	$26.09_{0.60}$	$34.55^{*}_{0.57}$	$57.9^{*}_{2.44}$	$\mathbf{59.56}^{*}_{1.50}$	$\underline{26.09}^{*}_{0.27}$	$\underline{28.35}^{*}_{0.28}$	$35.01^{*}_{0.22}$	$43.32^{*}_{0.27}$	34.63^{*} / 38.79^{*}
↪ + *BoN-Oracle*†	$47.90_{0.30}$	$44.19_{0.80}$	$30.07_{0.06}$	$42.78_{0.11}$	$62.07_{0.11}$	$68.62_{0.19}$	$36.16_{0.69}$	$38.87_{0.47}$	$48.78_{0.89}$	$49.71_{2.21}$	44.99 / 48.83
↪ + *BoN-ORM*	$37.73^{*}_{0.26}$	$37.94^{*}_{0.38}$	$\underline{26.74}^{*}_{0.18}$	$35.18^{*}_{0.59}$	$59.33^{*}_{0.12}$	$\underline{59.46}^{*}_{0.37}$	$26.73^{*}_{0.29}$	$28.54^{*}_{0.46}$	$34.83^{*}_{0.20}$	$42.68^{*}_{0.13}$	37.07^{*} / 40.76^{*}
LM-LEXICON-MOE (4×8B)											
↪ + *Zero-shot*	$\underline{40.09}^{*}_{0.12}$	$\underline{40.51}^{*}_{0.28}$	$23.35_{0.25}$	$32.94^{*}_{0.49}$	$\underline{60.31}^{*}_{0.55}$	$55.52_{0.33}$	$\mathbf{31.26}^{*}_{0.85}$	$\mathbf{33.81}^{*}_{2.26}$	$\underline{45.69}^{*}_{1.25}$	$\underline{46.07}^{*}_{1.06}$	$\underline{40.14}^{*}$ / $\underline{41.77}^{*}$
↪ + *BoN-Oracle*†	$47.39_{0.16}$	$40.31_{0.23}$	$30.87_{0.24}$	$43.24_{0.25}$	$51.62_{1.14}$	$61.88_{0.30}$	$35.23_{0.42}$	$35.69_{0.26}$	$54.84_{0.12}$	$50.50_{0.11}$	43.99 / 46.32
↪ + *BoN-ORM*	$\mathbf{40.33}^{*}_{0.18}$	$40.69^{*}_{0.26}$	$24.18_{0.37}$	$33.79^{*}_{0.64}$	$\mathbf{60.88}^{*}_{0.55}$	$57.66_{0.73}$	$\underline{31.08}^{*}_{0.17}$	$\underline{33.26}^{*}_{0.22}$	$\mathbf{45.86}^{*}_{0.38}$	$\mathbf{46.38}^{*}_{0.26}$	$\mathbf{40.46}^{*}$ / $\mathbf{42.35}^{*}$

Table 2: Main results on five benchmarks[6]. We highlight the **highest scores** among LM-LEXICON and compared methods; * denotes the significance test, where $p < 0.005$ between our method and Rerank-T5 (prior SoTA). ♣ denotes that we reproduce the in-distribution results with supervised training, and † indicates that the lines of results are not directly comparable with other settings. All **-ICL* settings employ the best setting with a 32-shot in practice.

Competitive Performance of LM-LEXICON. Table 2 presents the performance comparisons among baselines and existing SoTA methods for DM, including LM-LEXICON-DENSE models (trained on four real-world datasets) and LM-LEXICON-MOE, the proposed MoE model. LM-LEXICON outperforms strong supervised methods and frontier models with a distinct advantage. Specifically, (1) LM-LEXICON obtains nearly 10% extra BLEU and ROUGE improvements on 3D-EX over the prior SoTA. (2) It performs exceptionally on smaller datasets as well, for example, LM-LEXICON achieves the highest scores ($\{31.26\%, 33.81\%\}$ on {BLEU, ROUGE}) among all compared methods on Urban dataset, indicating the efficacy of our method to model rare word senses and usages. (3) The comparison between the many-shot learning of best perfomant frontier LMs and LM-LEXICON demonstrates that our method surpasses significantly larger dense models, by $\{23.44\%, 9.14\%\}$ on {Wiki, WordNet} in BLEU for instance. (4) It is also observed that the Oxford dataset has lower performance with our method. A possible reason is that a short term and relatively long context in Oxford makes it harder for the model to predict accurate definitions. Furthermore, compared to other benchmarks, the Oxford dataset exhibits a significantly high term overlap rate of around 80% along with a near-zero term-definition overlap rate. This stark contrast underscores the strong polysemy inherent in Oxford's terms. Consequently, models trained on Oxford struggle to generalize effectively when encountering previously seen terms used in different contexts. Overall, LM-LEXICON shows a clear advantage that confirms the effectiveness of introduced semantic expert specialization and semantic-focused sparsifying upcycling into LM-LEXICON.

& LM-LEXICON to extract our focused part of the generation.

Human Evaluation. The human evaluation was conducted using a random subset of 300 samples from the 3D-EX, comparing definitions generated by our model (LM-LEXICON-MOE) and the baselines (LM-LEXICON-DENSE and three proprietary models). We focus on comparing with proprietary models as they represent the current state-of-the-art in practical deployment and are the primary competitors in real-world lexicon construction scenarios. To obtain a fine-grained understanding of model-specific characteristics, we further propose five criteria: (1) *accuracy* measures how correctly the definition captures the core semantic meaning of the word; (2) *clarity* evaluates the definition's comprehensibility and transparency in conveying meaning, focusing on how easily readers can understand the concept; (3) *conciseness* assesses whether the definition achieves optimal length without re-

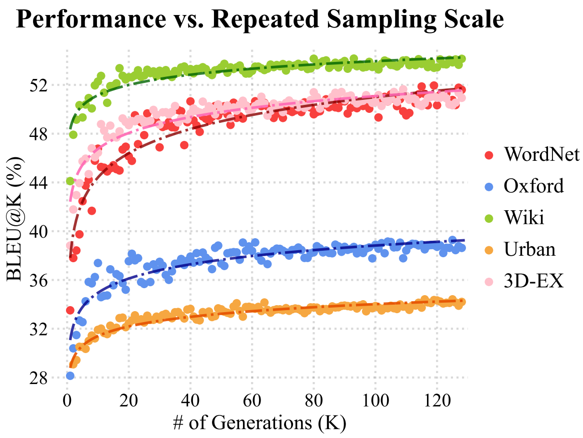

Figure 4: *Best-of-N* repeated sampling results (BLEU) on five benchmarks evaluated by **oracle verifier**.

dundancy or omission; (4) *context appropriateness* measures how well the definition reflects associated contexts, situations, and pragmatic constraints of the words; (5) *grammar and fluency* evaluates the grammatical correctness and naturalness of the definition. We employ three graduate students majoring in linguistics and lexicography, who were instructed to assess each of the above criteria on a 5-point scale, where 1 indicates the poorest quality and 5 represents the highest quality (Figure 12). The model names were kept anonymous from human evaluators to avoid possible bias, whereas the reference definitions remained accessible to them.

Figure 5 (right) presents the human evaluation results across five criteria, showing the average scores for each model[7]. LM-LEXICON-MOE consistently outperforms other models in most dimensions, with particularly strong performance of accuracy (4.6). While all models demonstrate competent performance with scores above 3.8, LM-LEXICON-MOE shows notable advantages in capturing contextual nuances and maintaining clarity and conciseness in definitions. The proprietary models perform similarly well but show slightly lower scores in terms of context appropriateness and conciseness than other criteria. We provide a detailed analysis of a representative example "*coon*" in Appendix E.

4.3 Ablation Study and Extra Investigation

In this section, we further conduct an in-depth analysis of LM-LEXICON, regarding: (1) data partition method, (2) routing policy, and (3) number of experts. In addition, we explore the impact of test-time scaling.

[7]Details on annotators' agreement can be found in §D.

Ablation on Different Data Partition Designs. Since LM-LEXICON integrates the knowledge acquired by experts from various data partitions, our first focus is on the impact of data partition methods. To this end, we considered three settings: (1) no split; (2) random split; and (3) lexical split. For random split, we follow Li et al. (2022) to slice the data into four balanced subsets and specialise an expert for each of them. For lexical split, we perform partition by TF-IDF (Sparck Jones, 1972).

As shown in Table 3, we observed that the original setting with semantic embedding clustering outperforms lexical-based partition with about $+7\%$ gains in BLEU and $+1\%$ gains in ROUGE on 3D-EX. The results imply that learning from semantic-targeted data clusters may help capture more precise senses and use more appropriate words to compose definitions. Lastly, it enables LM-LEXICON to develop more robust experts for various domains.

Model	BLEU	ROUGE	p-value
LM-LEXICON	$45.69_{\pm 0.3}$	$46.07_{\pm 0.1}$	–
+ w/ no split	$35.13_{\pm 0.2}$	$43.46_{\pm 0.3}$	$2.9e^{-5}$
+ w/ random split	$36.24_{\pm 1.4}$	$43.58_{\pm 0.8}$	$1.6e^{-5}$
+ w/ lexical split	$38.13_{\pm 0.5}$	$44.12_{\pm 0.6}$	$1.3e^{-4}$

Table 3: Ablation on data partition method.

Comparison among Routing Policies. Other than domain-level routing used in LM-LEXICON as default, we experiment on (1) top-1 token-level; (2) top-2 token-level; and (3) sequence-level routing. For token-level routing, we follow the implementation of Fedus et al. (2022) and Jiang et al. (2024). For sequence-level routing, we follow Pham et al. (2023).

Model	BLEU	ROUGE	p-value
LM-LEXICON	$45.69_{\pm 0.3}$	$46.07_{\pm 0.1}$	–
+ w/ top-1 token-level	$43.12_{\pm 0.4}$	$43.79_{\pm 0.5}$	$1.9e^{-3}$
+ w/ top-2 token-level	$45.38_{\pm 0.2}$	$45.21_{\pm 0.1}$	$8.6e^{-1}$
+ w/ sequence-level	$44.47_{\pm 0.2}$	$44.82_{\pm 0.3}$	$2.7e^{-3}$

Table 4: Ablation on different routing policies.

Table 4 presents that the domain-level routing (LM-LEXICON) is the most effective, even surpassing one of the popular scheme, the top-2 token-level routing, indicating that semantic routing via specified domain cluster is more beneficial for semantic-intensive tasks.

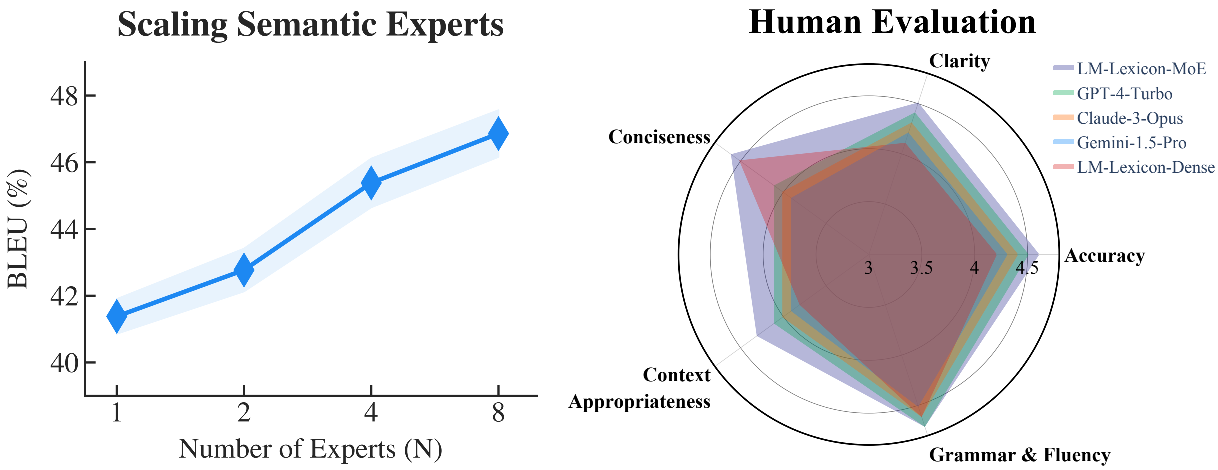

Figure 5: Scaling performance gains and human evaluation results. The left figure: Scaling test performance on 3D-EX, with varying number of experts. The right figure: Human evaluation results across five criteria.

Different Number of Semantic Experts. Except for the above four-experts LM-LEXICON-MOE, to investigate the impact of the number of semantic experts, we compare varied number of semantic experts ($N = 1, 2, 4, 8$). Notably, when $N = 1$, LM-LEXICON collapses back to a dense model and expands to a sparse model with $N > 1$ experts.

As shown in Figure 5 (left), we find that across all settings of N, the performance of our method consistently increases and outperforms the others, which are composed of fewer experts. For example, the model of $N = 1$ returns 41.38% while $N = 8$ yields 46.86% in BLEU. This tendency implies the scalability of our method, using more semantic experts. This trend can be extended by integrating more fine-grained semantic experts (Dai et al., 2024), but we leave this direction for future work.

Impact of Test-time Scaling. In light of Stiennon et al. (2020); Cobbe et al. (2021), we are curious on how to boost performance further via test-time scaling, notably ground truth-based (Oracle) verifier and Best-of-N (BoN) sampling with an outcome reward model (ORM). For oracle verifier, it uses reference as verification to provide binary feedbacks. For an ORM, it employs scalar feedback to select the optimal generation from candidates.

As depicted in Table 2 (BoN-ORM), interestingly, the oracle verifier is able to boost task performance (avg. ΔBLEU $> 2\%$) for LM-LEXICON-DENSE. However, it exhibits more limitations for LM-LEXICON-MOE; we speculate it is due to the diversity diminishment of models, as illustrated in Brown et al. (2024). Intuitively, optimal results are achieved with oracle verifier (Fig. 4) through repeated sampling with 128 completions per test sample. Intergating with the ORM or Oracle verifier, LM-LEXICON's generation quality shows consistent improvements across five benchmarks with the increase in the number of generations. This outcome aligns with the findings on math reasoning tasks (Cobbe et al., 2021; Brown et al., 2024).

5 Conclusion

In this paper, we present LM-LEXICON, an approach that combines domain experts upcycling with a sparse MoE model, which can generate appropriate definitions of terms in various domains and genres. We show that LM-LEXICON significantly outperforms frontier LLMs and strong supervised baselines. We hope LM-LEXICON could be extended to more domains and other semantic-intensive tasks in the future.

Limitations

Extrapolation to More Tasks. While we believe our observations and conclusions are comprehensive within our experimental settings, our work only focus on the task of definition modeling in English in this work. Future work could benefit from our findings in extending to other domains and related tasks in semantic-intensive scenarios.

Training Efficiency and Cost. Our method performs supervised fine-tuning of $N \times \mathcal{M}$ expert LMs that are initialized from a seed model. The training process can be thoroughly offline and asynchronous; however, it still needs an essential and sufficient computation budget to some extent. We encourage people to further explore parameter-efficient training methods based on LM-LEXICON.

Stronger Verifier. Our results from Section §4.3 highlight the importance of improving sample verification methods tailored for definition modeling, and even more general language generation, which are currently unavailable. Most existing verification methods have been developed only to solve complex reasoning tasks, such as mathematical, programming, and logical reasoning problems. We believe that equipping models with the ability to assess their own generations will allow test-time compute methods to be scaled further.

Ethics Statement

This research was conducted with careful consideration of ethical implications. All data used in this study was collected from public sources with appropriate permissions. We have taken measures to ensure privacy protection and prevent misuse of our model. The computational resources were used responsibly, and we have documented all potential biases and limitations. Our annotation process followed fair labor practices with appropriate compensation for annotators.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Thomas E. Ahlswede. 1985. A tool kit for lexicon building. In *23rd Annual Meeting of the Association for Computational Linguistics*, pages 268–276, Chicago, Illinois, USA. Association for Computational Linguistics.

Fatemah Almeman, Hadi Sheikhi, and Luis Espinosa Anke. 2023. 3D-EX: A unified dataset of definitions and dictionary examples. In *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing*, pages 69–79, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

Fatemah Yousef Almeman, Steven Schockaert, and Luis Espinosa Anke. 2024. WordNet under scrutiny: Dictionary examples in the era of large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 17683–17695, Torino, Italia. ELRA and ICCL.

AI Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, and 6 others. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Damai Dai, Chengqi Deng, Chenggang Zhao, R.x. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y.k. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. 2024. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1280–1297, Bangkok, Thailand. Association for Computational Linguistics.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. Toy models of superposition. *Transformer Circuits Thread*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

J. L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

A Gadetsky, I Yakubovskiy, and D Vetrov. 2018. Conditional generators of words definitions. In *ACL 2018-56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, pages 266–271.

Mario Giulianelli, Iris Luden, Raquel Fernandez, and Andrey Kutuzov. 2023. Interpretable word sense representations via definition generation: The case of semantic change analysis. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3130–3148, Toronto, Canada. Association for Computational Linguistics.

Suchin Gururangan, Margaret Li, Mike Lewis, Weijia Shi, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. 2023. Scaling expert language models with unsupervised domain discovery. *arXiv preprint arXiv:2303.14177*.

Lotte Hogeweg and Agustin Vicente. 2020. On the nature of the lexicon: The status of rich lexical meanings. *Journal of Linguistics*, 56(4):865–891.

Han Huang, Tomoyuki Kajiwara, and Yuki Arase. 2021. Definition modelling for appropriate specificity. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2499–2509, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shonosuke Ishiwatari, Hiroaki Hayashi, Naoki Yoshinaga, Graham Neubig, Shoetsu Sato, Masashi Toyoda, and Masaru Kitsuregawa. 2019. Learning to describe unknown phrases with local and global contexts. 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 3467–3476, Minneapolis, Minnesota. Association for Computational Linguistics.

James Jhirad, Edison Marrese-Taylor, and Yutaka Matsuo. 2023. Evaluating large language models' understanding of financial terminology via definition modeling. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 93–100.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Cunliang Kong, Yun Chen, Hengyuan Zhang, Liner Yang, and Erhong Yang. 2022. Multitasking framework for unsupervised simple definition generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5934–5943, Dublin, Ireland. Association for Computational Linguistics.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech Republic. Association for Computational Linguistics.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2025. Nv-embed: Improved techniques for training llms as generalist embedding models. *Preprint*, arXiv:2405.17428.

Leeroo-AI. 2024. Mergoo: A library for easily merging multiple llm experts, and efficiently train the merged llm. `https://github.com/Leeroo-AI/mergoo`. Accessed: 2024-07-23.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. 2022. Branch-train-merge: Embarrassingly parallel training of expert language models. *arXiv preprint arXiv:2208.03306*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

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

Mikko I. Malinen and Pasi Fränti. 2014. Balanced k-means for clustering. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 32–41, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ke Ni and William Yang Wang. 2017. Learning to explain non-standard english words and phrases. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 413–417.

Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. 2017. Definition modeling: Learning to define word embeddings in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and 1 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Francesco Periti, David Alfter, and Nina Tahmasebi. 2024. Automatically generated definitions and their utility for modeling word meaning. In *Proceedings of the 2024 Conference on Empirical Methods in*

Natural Language Processing, pages 14008–14026, Miami, Florida, USA. Association for Computational Linguistics.

Savvas Petridis, Ben Wedin, Ann Yuan, James Wexler, and Nithum Thain. 2024. ConstitutionalExperts: Training a mixture of principle-based prompts. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 574–582, Bangkok, Thailand. Association for Computational Linguistics.

Hai Pham, Young Jin Kim, Subhabrata Mukherjee, David P. Woodruff, Barnabas Poczos, and Hany Hassan. 2023. Task-based MoE for multitask multilingual machine translation. In *Proceedings of the 3rd Workshop on Multi-lingual Representation Learning (MRL)*, pages 164–172, Singapore. Association for Computational Linguistics.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. Mauve: Measuring the gap between neural text and human text using divergence frontiers. *Advances in Neural Information Processing Systems*, 34:4816–4828.

James Pustejovsky and Branimir Boguraev. 1993. Lexical knowledge representation and natural language processing. *Artificial Intelligence*, 63(1):193–223.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Zhihong Shao, Damai Dai, Daya Guo, Bo Liu, and Zihan Wang. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *ArXiv*, abs/2405.04434.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

Zhengyan Shi, Adam X Yang, Bin Wu, Laurence Aitchison, Emine Yilmaz, and Aldo Lipani. 2024. Instruction tuning with loss over instructions. *arXiv preprint arXiv:2405.14394*.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, volume 33, pages 3008–3021. Curran Associates, Inc.

Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Roziere, Jacob Kahn, Shang-Wen Li, Wen tau Yih, Jason E Weston, and Xian Li. 2024. Branch-train-mix: Mixing expert LLMs into a mixture-of-experts LLM. In *First Conference on Language Modeling*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and 1 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Zhenyu Wu, Yaoxiang Wang, Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Jingjing Xu, and Yu Qiao. 2023. OpenICL: An open-source framework for in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 489–498, Toronto, Canada. Association for Computational Linguistics.

Yunting Yin and Steven Skiena. 2023. Word definitions from large language models. *arXiv preprint arXiv:2311.06362*.

Hengyuan Zhang, Dawei Li, Shiping Yang, and Yanran Li. 2022. Fine-grained contrastive learning for definition generation. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1001–1012.

Linhan Zhang, Qian Chen, Wen Wang, Yuxin Jiang, Bing Li, Wei Wang, and Xin Cao. 2023. Exploiting correlations between contexts and definitions with multiple definition modeling. *arXiv preprint arXiv:2305.14717*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M Meyer, and Steffen Eger. 2019. Moverscore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings* of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578.

Yuhang Zhou, Giannis Karamanolakis, Victor Soto, Anna Rumshisky, Mayank Kulkarni, Furong Huang, Wei Ai, and Jianhua Lu. 2025. MergeME: Model merging techniques for homogeneous and heterogeneous MoEs. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2315–2328, Albuquerque, New Mexico. Association for Computational Linguistics.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. 2024. LLaMA-MoE: Building mixture-of-experts from LLaMA with continual pre-training. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15913–15923, Miami, Florida, USA. Association for Computational Linguistics.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. 2022. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*.

A Additional Experiment Details

This is a section in the appendix. Introduce dataset components, hyperparameter settings, and other experimental details.

Data Processing. Raw 3D-EX (see fig. 6) consists of ten lexicon sources of $<t, c, d>$ triplets, we use the word-level split on each of the sources to train, validate and test our models in this paper. We developed the following steps to undergo the preprocessing procedure for the raw 3D-EX dataset.

- We filter out all instances from the subsets including Hei++, MultiRD, and Webster's Unabridged, since they do not have any usable example context for each term of words.
- We discard instances that do not meet any of the following conditions: ① TERM must be of string type, ② DEFINITION must be of string type, ③ EXAMPLE must not be empty, and ④ DATASET_NAME must not be empty.
- To enhance the model's ability to interpret words in various contexts, we split the sample entries with multiple example contexts into separate data instances for each context. This approach increases the number of samples the model sees during training.

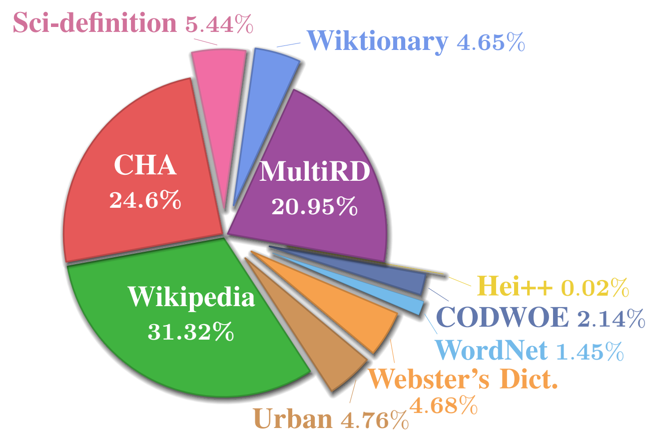

Figure 6: 3D-EX constituents distribution.

In addition, we observed many examples in the existing datasets that share the same term-context pair but with different definitions, which may cause negative effects on model learning if there exist many semantics-divergent examples. To summarize and display the potential impacts, we report the salient statistics about this finding of these datasets shown in the following Table 5.

Dataset	Split	# All	# Div.	% Div. / All
WordNet	$\mathcal{S}_{\text{train}}$	13,883	2,723	19.61
	$\mathcal{S}_{\text{valid}}$	1,752	368	21.00
	$\mathcal{S}_{\text{test}}$	1,775	333	18.76
Oxford	$\mathcal{S}_{\text{train}}$	82,479	34	0.04
	$\mathcal{S}_{\text{valid}}$	10,285	2	0.02
	$\mathcal{S}_{\text{test}}$	10,306	0	0.00
Wikipedia	$\mathcal{S}_{\text{train}}$	887,455	186	0.02
	$\mathcal{S}_{\text{valid}}$	44,003	16	0.04
	$\mathcal{S}_{\text{test}}$	57,232	14	0.02
Urban	$\mathcal{S}_{\text{train}}$	411,382	1,424	0.35
	$\mathcal{S}_{\text{valid}}$	57,883	152	0.26
	$\mathcal{S}_{\text{test}}$	38,371	122	0.32
3D-EX	$\mathcal{S}_{\text{train}}$	1,309,312	35,632	2.72
	$\mathcal{S}_{\text{valid}}$	513,789	12,551	2.44
	$\mathcal{S}_{\text{test}}$	450,078	7,599	1.69

Table 5: Divergent examples statistics of each dataset. **# All**: number of all examples; **# Div.**: number of all divergent examples; **% Div. / All**: ratio of divergent examples in all examples.

Clustering Setup. Compared with Gururangan et al. (2023), we consider to mine the intrinsit semantic meaning of term associated with their context, instead of using lexical statistics clustering method, like TF-IDF. We argue that the method building on dense semantic clustering would help upcycling models to learn specialized sense interpretation-oriented experts, towards robust system for definition modeling. We run k-means++ clustering of the Elkan variation method with $1,000$ max iteration, $1e^{-8}$ tolerance of convergence, and a fixed seed of 42. Considering the computation and memory bounds, we first use 4 as the number of clusters to form and the number of centroids to generate. We further ablate this factor in the section §4.3.

Training Details. LM-LEXICON was trained for 3 epochs with a global batch size of 8,192 tokens (gradient accumulation 1, batch size per device 8, max sequence length 128) on 8 × H100-PCIe-80GB GPUs and a learning rate of 1e-6, minimum learning rate of 3e-7 with a cosine annealing scheduler, as well as the warm-up steps with 6% ratio of the total training steps. We used a global dropout of 0.2 (Srivastava et al., 2014) and a weight decay of 0.1 with AdamW optimizor (Loshchilov and Hutter, 2018), and performed early stopping to obtain the best model by the highest validation bleu.

Moreover, We run three times for each training setup to report the mean results and their standard deviation of metrics, with seed $s_i \in \{21, 42, 84\}$, respectively. We use Hugging Face Transformers (Wolf et al., 2020) and Pytorch (Paszke et al., 2019) to develop the training pipeline.

We run the branch training on each cluster of data points obtained from the clustering results. As depicted in tab. 12, We set up the following hyper-parameters to train LM-LEXICON and vanilla fine-tuned LLAMA-3-8B models in this paper. We used the standard negative log-likelihood (NLL) loss to train LM-LEXICON. Contrary to Shi et al. (2024), to avoid the loss of the input sequence tokens overshadowing the actual output token loss, the loss is only computed over the result tokens (Eq. 1), limiting the potential to overfit to the input prompt and context. This loss calculation method resulted in faster training and robuster results overall.

Given a definition generation problem $p(c, t)$ and its golden reference d, we define a outcome reward model as the following: ORM ($P \times D \rightarrow \mathbb{R}$) assigns a single value to s to indicate whether predicted $\hat{d}$ is correct. Given a specific dataset $\mathcal{D}$, we follow Cobbe et al. (2021) to use a negative log-likelihood loss (Eq. 3) to frame the reward modeling as a binary classification objective.

$$\mathcal{L}_{\text{ORM}} = -\log \sigma \left(r_\phi(x, y_w) - r_\phi(x, y_l) \right) \quad (3)$$

Where y_w is the preferred generation (i.e., chosen response) and y_l is the alternate generation (i.e., rejected response) conditioned on the input $x := p(c, t)$. To train a ORM built on training set, we leverage the golden reference d as the preferred definition y_w and one of the model generations as the alternate definition y_l to express preferences for each x, denoted as $y_w \succ y_l \mid x$, where y_w and y_l denotes the preferred and dispreferred completion, respectively. σ is the sigmoid function and $r_\phi(\cdot, \cdot)$ represents the parameterized reward function for the concatenated input x and generation y_*. To enhance computing efficiency, we employ the ratio of $1 : 32$ to conduct repeated sampling and rerank the generations by their log-likelihood (aka. confidence) to acquire the top-eight items as a candidate set of alternate generations for each input x.

Inference Setup. As shown in Table 2, for each setting in "Zero-shot", "BoN-Oracle", and "BoN-ORM", we orchestrate three separate runs for each setting, using the same decoding parameters but with different random seeds to ensure robustness and consistency in the results. Specifically, for the models LM-LEXICON-DENSE and LM-LEXICON-MOE, specifically, we use the temperature of 0.6, *top-k* of 50, *top-p* of 0.9, and repetition penalty of 1.05, ensuring uniformity across all evaluations.

For all benchmarks included in our test, as the number of samples increases, the coverage metric corresponds to the use of an oracle verifier. This verifier checks which fraction of DM problems in the test set can be approximated using any of the samples that were generated to be as similar as possible to the ground truth. The selection of the most similar generation is achieved through an iterative comparison with the golden definition, ensuring a robust matching process. In the case of the oracle verification process by the oracle verifier, we validate whether any output chosen prediction is the most similar by comparing it with golden references of the sample in the test set. In contrast, for the verification process of ORM verifier, the selection of the most similar generation is then performed solely by the ORM verifier itself, without relying on external feedback, ground-truth comparison, or oracle input.

Miscellaneous. We developed our MoE language modeling codebase based on Leeroo-AI (2024) and implemented several routing policies and proposed MoE architectures. Aiming at more efficent evlauation, we follow (Huang et al., 2021) and refactor their implementation with concurrent metrics computation to boost the inference procedure in large models, please see the details in our released code.

B Carbon Footprint

The cost of fine-tuning LLM is lower than that of pre-training them. Nevertheless, we think it is critical to quantify and record the environmental consequences of our research. Table 6 lists the materials required for a single run, which is conducted using our own infrastructure. We calculate the carbon footprint estimation using a carbon intensity of 0.141 kg/kWh and 700W consumption per GPU[8].

Model	Hardware	FLOPs	Time (h)	CO2eq (kg)
LM-Lexicon-Dense	8×H100	$4.2e^{18}$	36.4	11.4
LM-Lexicon-MoE	8×H100	$5.4e^{18}$	32.8	14.6

Table 6: Details about the training required resources.

C Additional Evaluation Results

C.1 Data Clustering Results

Cluster C_i	Distance$_{\text{intra-cluster}}$ ↓
C_0 (Adjective)	0.176
C_1 (Scientific)	0.168
C_2 (Proper Noun)	0.173
C_3 (Person Name)	0.185
Average	0.175

Table 7: Intra-cluster Distances (*i.e.,* the cluster cohesion)

.

We show the clustering results including cluster cohesion and cluster separation in the following Table 7 and 8, respectively.

C.2 In-Context Learning Evaluation

We show the scaling in-context learning experimental results as shown in Figure. 7.

C.3 Generation Examples of LM-Lexicon

As depicted in Figure 8, 9, 10, and 11, we provide a cherry-picked example for each domain cluster as shown in Figure 3 in definition modeling.

[8]Statistics: `https://app.electricitymaps.com/map`.

Cluster (C_i, C_j)	Distance$_{\text{inter-cluster}}$ ↑
C_0, C_1	0.694
C_0, C_2	0.713
C_0, C_3	0.765
C_1, C_2	0.681
C_1, C_3	0.707
C_2, C_3	0.720
Average	0.713

Table 8: Inter-cluster Distances (*i.e.,* the cluster separation): C_0 denotes the domain of "Adjective", C_1 denotes the domain of "Scientific", C_2 denotes the domain of "Proper Noun", and C_3 denotes the domain of "Person Name".

Cluster-1 Example:

[Term] **Combtooth Blenny**

[Query] "the crested blenny is a species of **Combtooth Blenny** found around New South Wales, Australia, ..." What is the definition of "**Combtooth Blenny**"?

[Source] Wikipedia

[Reference] **Combtooth Blenny**: perciform marine fish of the family blenniidae.

Figure 8: Example of $\mathcal{C}_1$ (proper noun) from 3D-EX.

Cluster-2 Example:

[Term] **brave**

[Query] "familiarity with danger makes a **brave** man braver but less daring - herman melville ..." What is the definition of "**brave**"?

[Source] WordNet

[Reference] **brave**: possessing or displaying courage; able to deal with danger or fear without flinching.

Figure 9: Example of $\mathcal{C}_2$ (adjective) from 3D-EX.

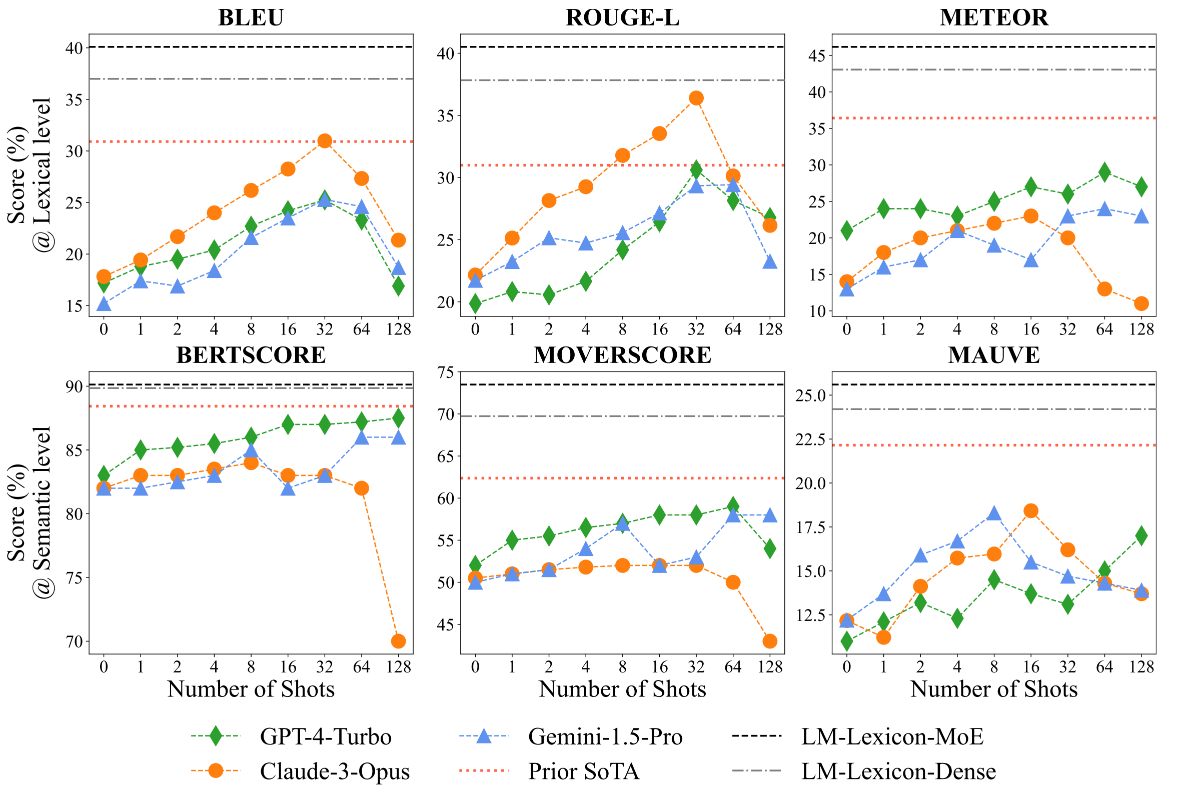

Figure 7: Scaling the in-context learning results of frontier causal LMs on WordNet with k-shot demonstrations, where k scales logarithmically from 0 to 128. Prior SoTA denotes the Rerank-T5 proposed by Huang et al. (2021).

Cluster-3 Example:

[Term] **Michael Maclennan**

[Query] "Godiva's is a Canadian television comedy-drama series created by **Michael Maclennan** with Julia Keatley of Keatley Entertainment ..." What is the definition of "**Michael Maclennan**"?

[Source] Wikipedia

[Reference] **Michael Maclennan**: Canadian playwright, screenwriter, and producer of television shows.

Figure 10: Example of $\mathcal{C}_3$ (person name) from 3D-EX.

Cluster-4 Example:

[Term] **Lymphedema-distichiasis Syndrome**

[Query] "two patients with **Lymphedema-distichiasis Syndrome** illustrate that both Milroy's ..." What is the definition of "**Lymphedema-distichiasis Syndrome**"?

[Source] Sci-definition

[Reference] **Lymphedema-distichiasis Syndrome**: lymphedema distichiasis syndrome is a condition that affects the normal function of the lymphatic system.

Figure 11: Example of $\mathcal{C}_4$ (scentific) from 3D-EX.

D Human Evaluation Agreement

To assess the agreement among the annotators, we employed Fleiss's Kappa (Fleiss, 1971), which is a statistical measurement to assess the reliability of the agreement between multiple raters. Fleiss's Kappa account for the possibility of agreement occurring by chance. It is calculated using the following formula:

$$\kappa = \frac{P_o - P_e}{1 - P_e}$$

where:

- P_o is the observed agreement among the raters, and
- P_e is the expected agreement by chance.

Table 9 presents Fleiss's Kappa coefficients for human evaluation agreement on each criterion and model.

Criteria	LM-Lexicon-MoE	LM-Lexicon-Dense	Claude-3-Opus	Gemini-1.5-Pro	GPT-4-Turbo
Accuracy	0.85	0.78	0.80	0.79	0.77
Clarity	0.82	0.75	0.78	0.76	0.74
Conciseness	0.80	0.74	0.77	0.75	0.73
Context Appropriateness	0.82	0.76	0.79	0.78	0.76
Grammar & Fluency	0.78	0.76	0.75	0.74	0.73

Table 9: Fleiss's Kappa coefficient statistics for the agreement of human evaluation on each criterion and model.

E Comparison of Different Definitions

The following is a representative case of the generated definitions from five models including three proprietary models, LM-Lexicon-Dense and LM-Lexicon-MoE:

Word: "coon"

Context: "I'll be a gone coon when the battle starts"

Reference: "an eccentric or undignified rustic" (from WordNet)

In the demonstration in Table 10, definition generated by LM-Lexicon-MoE (our method) is most closely aligned with the reference definition's core meaning, which captures the derogatory connotation and reflects the "undignified" aspect. In contrast, definitions produced by the other models fail to capture fundamental aspects of the reference definition, for instance, Claude-3-Opus and GPT-4-Turbo completely deviate from the basic meaning of the target word in the context.

In general, there are some notable patterns:

- Closed-source models (e.g., GPT-4 and Claude) generate overly verbose definitions, including redundant contextual information and multiple interpretations.
- These models tend to overemphasize derivative or secondary meanings, resulting in broad, general definitions that deviate from the core semantic meaning.
- Models trained on general corpora often struggle with uncommon word usages due to limited training examples, leading to inaccurate definitions or contextual misinterpretation.

This comparison demonstrates the superiority of LM-Lexicon-MoE's targeted, controlled approach over general-purpose language models for definition modeling.

F Code for LM-Lexicon

We use the Alg. 2 and 3 provided below to train LM-Lexicon-MoE on the 3D-EX and the other four legacy datasets used in this paper. We exposed this Pytorch-style code as an implementation to extend our method to any potential domain. Additionally, to illustrate the differences and novelty of our method, we provide a comparison with the relative methods as shown in Table 11.

Model	**Generated Definition**
Claude-3-Opus	The phrase gone coon means someone who is certainly defeated or about to die.
Gemini-1.5-Pro	The word “coon” is a deeply offensive racial slur targeting people of African descent. It has a long and painful history rooted in racist caricatures and stereotypes. While it once appeared casually in entertainment and everyday language, it is now considered completely unacceptable to use.
GPT-4-Turbo	“Coon” is a colloquial abbreviation for “raccoon,” a nocturnal mammal native to North America. “Coon” was sometimes used to refer to a person who is in deep trouble or doomed, often in the phrase “a gone coon,” meaning someone who is as good as dead or has no chance of survival.
LM-Lexicon-Dense (Ours)	A person who is afraid
LM-Lexicon-MoE (Ours)	A person who is deemed to be despicable or contemptible

Table 10: Comparison of generated definition by models.

	MoE (2017) (Vanilla)	BTM (2022) (Merge)	BTX (2024) (Linear router)	LM-Lexicon (Ours)
◇ Dense experts are trained independently (upcycling)	✘	✔	✔	✔
◇ Experts are specialized in different domains	✘	✔	✔	✔
◇ Experts are chosen by a learned router per input token	✔	✘	✔	✔
◇ Adaptive router via domain-wise routing	✘	✘	✘	✔
◇ Semantic experts adapted to diverse domains	✘	✘	✘	✔

Table 11: A comprehensive comparison of the most relative sparse mixture-of-experts frameworks in recent years, including MoE (Vanilla), BTM (Merge), BTX (Linear Router), and LM-Lexicon. Our method demonstrates advancements in semantic-centric specialized expert and adaptability across domains.

Algorithm 2 Pytorch code for semantic experts merger.

```python
def merge_semantic_experts(experts, router_layers):
    """
    Merge expert models into a unified model.

    Args:
        - experts (ModuleList): Experts to merge.
        - router_layers (ModuleList): Router layers.

    Returns:
        - state_dict (Dict[str, Tensor]): Merged model weights.
    """
    state_dict = dict()
    expert_nums = len(experts)
    count_total_router_layers = 0

    for idx, expert in enumerate(experts):
        # load each expert model
        model_id = expert["model_id"]
        model = load_base_model(model_id)

        if hasattr(model, "_tied_weights_keys"):
            tied_weights_keys.extend(model._tied_weights_keys)
            count_router_layers = 0
            count_averaged_layers = 0

        # iterate over all the layers of the model
        for layer_name, param in model.state_dict().items():
            is_merge_layer = True
            for router_layer in router_layers:
                if is_layer_suitable_for_router(router_layer, layer_name):
                    is_merge_layer = False
                    wb = layer_name.split(".")[-1]
                    new_layer_name = layer_name.split(f"{wb}")[0]
                    new_layer_name = f"{new_layer_name}experts.{ix}.{wb}"
                    assert new_layer_name not in state_dict
                    state_dict[new_layer_name] = param
                    count_total_router_layers += 1
                    count_router_layers += 1

            if is_merge_layer:
                # average the rest of layers by mean of weights
                prev_weight = state_dict.get(layer_name)

                if prev_weight is None:
                    prev_weight = torch.tensor(0)
                else:
                    if not prev_weight.shape == param.shape:
                        # adjust the shape of weight
                        prev_weight, param = shape_adjuster(
                            prev_weight, param, idx
                        )

                try:
                    # sometimes data is empty / non weights
                    state_dict[layer_name] = prev_weight + (param / expert_nums)
                except Exception as _:
                    print(layer_name, param)
                    state_dict[layer_name] = param

                count_averaged_layers += 1

    return state_dict
```

Algorithm 3 Pytorch code for modeling LM-LEXICON-MOE Layer

```
class SemanticMoeLayer(nn.Module):
    def __init__(
        self,
        in_features: int,
        out_features: int,
        bias: bool,
        num_experts: int,
        num_experts_per_tok: int = 2,
        routing_policy: str,
    ):
        """Semantic Mixture-of-Experts Layer.

        Args:
            - in_features (int): Input Features
            - out_features (int): Output Features
            - bias (bool): Use bias or not.
            - num_experts (int): Total numbers of experts that Router Layer would handle
            - num_experts_per_tok (int): Number of active experts per token.
            - routing_policy (str): Routing Policy.
        """
        super().__init__()
        self.routing_policy = routing_policy
        if routing_policy == "token-level":
            # top-k token-level routing
            self.gate = nn.Linear(in_features, num_experts, bias=False)
            self.experts = nn.ModuleList(
                [nn.Linear(in_features, out_features, bias) for _ in range(num_experts)]
            )
            self.num_experts_per_tok = num_experts_per_tok
            self.in_features = in_features
            self.out_features = out_features
        elif routing_policy in ["soft-sequence-level", "hard-sequence-level"]:
            # soft/hard sequence-level routing
            self.gate = nn.Linear(in_features, num_experts, bias=False)
            self.num_experts = num_experts
            self.experts = nn.ModuleList(
                [nn.Linear(in_features, out_features) for _ in range(num_experts)]
            )
        elif routing_policy == "domain-level":
            # domain-level routing
            self.gate = nn.Linear(in_features, num_experts, bias=False)
            self.num_experts = num_experts
            self.experts = nn.ModuleList(
                [nn.Linear(in_features, out_features) for _ in range(num_experts)]
            )

    def forward(self, inputs: torch.Tensor, domain_labels: torch.Tensor):
        if self.routing_policy == "token-level":
            gate_logits = self.gate(inputs)
            weights, selected_experts = torch.topk(
                gate_logits, self.num_experts_per_tok
            )
            weights = F.softmax(weights, dim=2, dtype=torch.float).to(inputs.dtype)
            results = torch.zeros(
                (inputs.shape[0], inputs.shape[1], self.out_features),
                device=inputs.device,
                dtype=inputs.dtype,
            )

    # continue this table as below ...
```

```python
# continue the above table ...

        weights = weights.to(inputs.device)
        for ix, expert in enumerate(self.experts):
            batch_idx, tok_idx, expert_idx = torch.where(selected_experts == ix)
            results[batch_idx, tok_idx] += expert(
                inputs[batch_idx, tok_idx]
            ) * weights[batch_idx, tok_idx, expert_idx].unsqueeze(-1)
    elif self.routing_policy == "soft-sequence-level":
        # soft sequence-level routing
        gate_logits = self.gate(inputs)
        gate_logits_mean = gate_logits.mean(dim=1)
        weights = F.softmax(gate_logits_mean, dim=-1)
        results = torch.zeros(
            (inputs.shape[0], inputs.shape[1], self.out_features),
            device=inputs.device,
            dtype=inputs.dtype,
        )
        for ix, expert in enumerate(self.experts):
            results += expert(inputs) * weights[:, ix].unsqueeze(-1)
    elif self.routing_policy == "hard-sequence-level":
        # hard sequence-level routing (only one selected expert is responsible for the
            entire sequence)
        gate_logits = self.gate(inputs)
        gate_logits_mean = gate_logits.mean(dim=1)
        _, selected_experts = torch.topk(gate_logits_mean, 1)
        results = torch.zeros(
            (inputs.shape[0], inputs.shape[1], self.out_features),
            device=inputs.device,
            dtype=inputs.dtype,
        )
        for ix, expert in enumerate(self.experts):
            results += expert(inputs) * (selected_experts == ix).float().unsqueeze(
                -1
            )
    elif self.routing_policy == "domain-level":
        # domain-level routing (only one selected expert is responsible for the entire
            sequence)
        gate_logits = self.gate(inputs)
        results = torch.zeros(
            (inputs.shape[0], inputs.shape[1], self.out_features),
            device=inputs.device,
            dtype=inputs.dtype,
        )
        for ix, expert in enumerate(self.experts):
            results += expert(inputs) * (domain_labels == ix).float().unsqueeze(-1)

    return results
```

Computing Infrastructure
8 × H100-80GB GPU (PCIe)

Hyperparameter	**Assignment**	**Hyperparameter**	**Assignment**
Base model	LM-Lexicon-Dense (Llama-3-8B)	Base model	LM-Lexicon-MoE (4 × Llama-3-8B)
Training strategy	DS ZERO-3	Training strategy	NAIVE PP
Epochs	3	Epochs	1
Global batch size	524,288 tokens	Global batch size	131,072 tokens
Max sequence length	128	Max sequence length	128
Max learning rate	$5e-6$	Max learning rate	$1e-6$
Optimizer	AdamW	Optimizer	AdamW
Adam beta weights	$0.9, 0.95$	Adam beta weights	$0.9, 0.95$
Learning rate schedule	Cosine decay to 0	Learning rate schedule	Cosine decay to 0
Weight decay	0.01	Weight decay	0.01
Warm-up ratio	10%	Warm-up ratio	10%
Gradient clipping	1.0	Gradient clipping	1.0
Global dropout	0.1	Global dropout	0.1
Random seeds	$\{21, 42, 84\}$	Random seeds	$\{21, 42, 84\}$

Table 12: Hyper-parameters of LM-LEXICON-DENSE and LM-LEXICON-MOE training. DS ZERO-3 (left-hand table) denotes stage-3 ZeRO parallelism implemented by DeepSpeed (Rajbhandari et al., 2020). NAIVE PP (right-hand table) denotes naive pipeline parallelism implemented by 🤗 Hugging Face Transformers (Wolf et al., 2020).

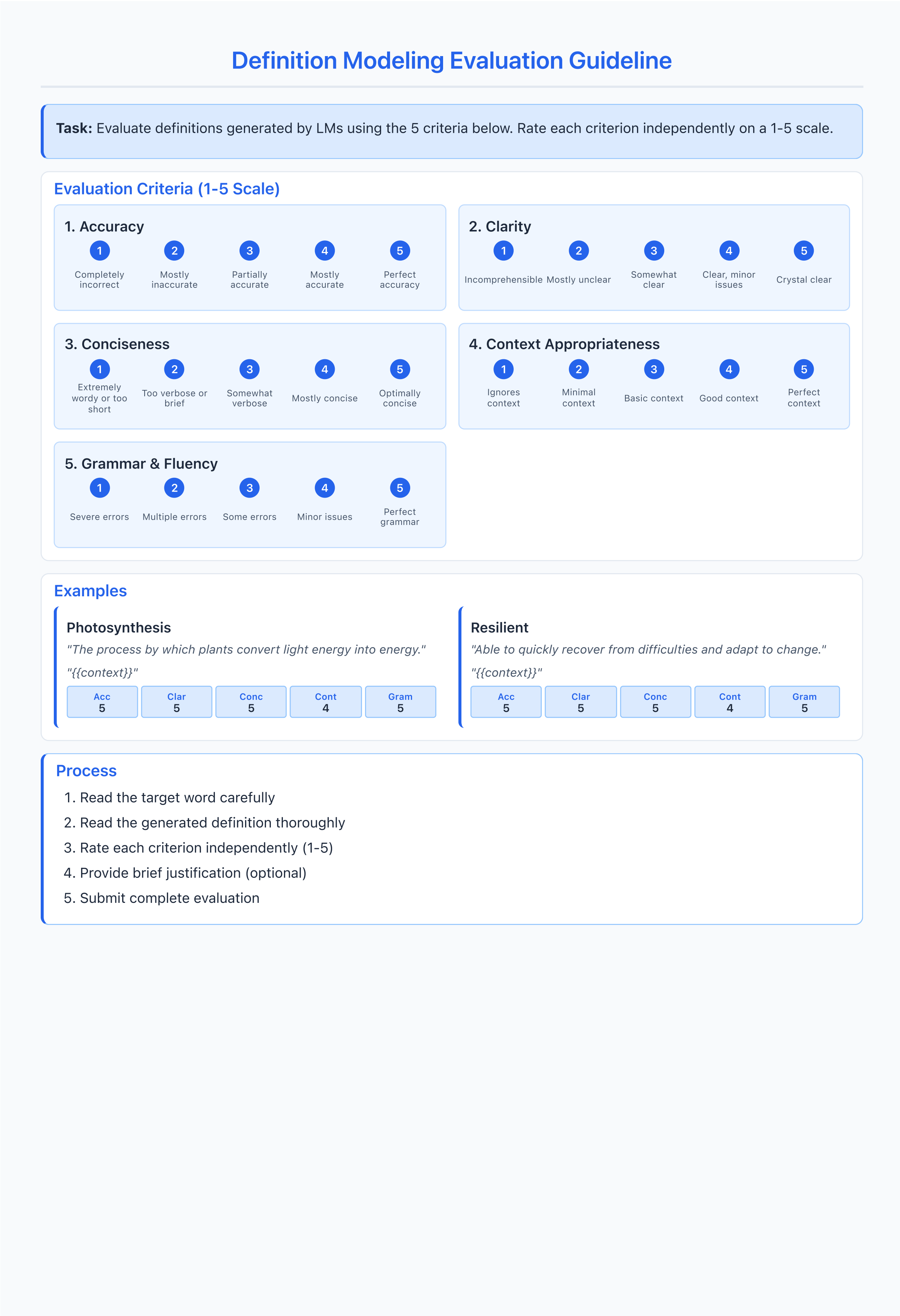

Figure 12: Human evaluation guideline.