LM-LEXICON: Definition Modeling with Mixture-of-Experts

Anonymous ACL submission

Abstract

We introduce LM-LEXICON, a 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 and targeted language models for semantic-intensive applications.

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

1 Introduction

Defining terms is the first step towards building a lexicon for a language (Pustejovsky and Boguraev, 1993). Precise definition should be formed as highly summarized and human-readable sentences that capture the major sense of the term. Conventionally, the labor involved in constructing such a lexicon manually is overwhelming (Ahlswede, 1985) and most lexicons are designed to omit words with usage limited to specific, isolated, or informal contexts (Michel et al., 2011). However, as illustrated in Fig.1, a lexicon needs to be updated to include new terms, novel senses, meaning shifts of existing terms, and domain knowledge (Hogeweg and Vicente, 2020). As such, definition modeling (DM) is a plausible solution to this problem where definitions are generated, conditioned, and changed diachronically on the target term and the context in which they occur (Hill et al., 2015; Noraset et al., 2017; Fedorova et al., 2024, *inter alia*).

While previous work on DM yields reasonable results, they fail to capture more subtle and rare senses, lacking semantic completeness (Huang et al., 2021; Giulianelli et al., 2023; Periti et al., 2024). Recently, acquiring definitions of terms using large language models (LLMs) has attracted increasing interest. LLMs can produce high-quality definitions comparable to the lexicon but often lead to under-specific and over-specific issues (Jhirad et al., 2023; Yin and Skiena, 2023; Almeman et al., 2024). Moreover, existing methods struggle in effectively modeling lexical resources across multiple domains and genres, due to their semantic heterogeneity and idiosyncrasy (Huang et al., 2021;

Zhang et al., 2022; Kong et al., 2022; Giulianelli et al., 2023). Hence, we argue that dense language model-based methods may not be able to handle the semantics in the over-superposition condition (Elhage et al., 2022; Bricken et al., 2023) and further predict definitions precisely for terms across various domains.

To mitigate these issues, we propose LM-LEXICON (**L**anguage **M**odel as **Lexicon**), which learns to perform definition modeling covering multiple domains, enabling the model to adapt diverse genres of definitions with a mixture-of-experts architecture. Unlike recent work (Sukhbaatar et al., 2024; Zhu et al., 2024a), our method efficiently combines MoE and data clustering together and obtains significant performance gains in DM. As shown in Figure 2, instead of directly fine-tuning on the raw definition corpora, our method allows the training of each expert in the branch and merging them by composing their specialized weights.

In summary, we contribute:

- LM-LEXICON-DENSE and LM-LEXICON-MOE: The best LMs for definition modeling.
- We introduce a new method for DM, where specialized semantic experts can be integrated for domain updates, enabling adaptation and generalization to new domains, or collapsing back to a single expert for efficient inference. Our method has the potential to be extended to underrepresented domains in the future, such as finance, law, and biomedicine.
- We conduct a comprehensive evaluation for testing current proprietary large language models in zero-, few-, and many-shot setups with diverse in-context learning strategies. Empirically, we unveil that these frontier large language models may struggle to generate appropriate definitions even with many-shot settings in a semantic-intensive scenario.

2 Related Work

Instruction Tuning (IT). LMs can better align with human intents through supervised instruction tuning (Wei et al., 2021; Sanh et al., 2021; Ouyang et al., 2022; Zhou et al., 2024, *iter alia*). IT has empowered many tasks in NLG such as machine translation (Li et al., 2024a; Zhu et al., 2024b; Pan et al., 2024), summarization (Fetahu et al., 2023; Pu et al., 2023; Zhang et al., 2024), and dialogue generation (Chen et al., 2023; Zheng et al., 2024; Yi et al., 2024), it has not widely used in DM task yet. As the first task-focusing study, Giulianelli et al. (2023) used FLAN-T5 with continual IT to conduct DM in multiple domains but focusing on performance gains in distribution shift across datasets.

Upcycling to Mixture-of-Experts (MoE). On the model efficiency and expressiveness aspect, Shazeer et al. (2017); Fedus et al. (2022); Jiang et al. (2024); Shao et al. (2024) focus on designing efficient sparse MoE architecture with token-level routing policy. On 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.

Definition Modeling (DM). 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, local, or both global and local contexts of a term, to generate definitions of given target. Then Huang et al. (2021); Kong et al. (2022); Zhang et al. (2022); Giulianelli et al. (2023); Periti et al. (2024) propose a series of methods for this task based on transformer seq2seq LMs (*e.g.*, T5) and Causal LMs. In the era of LLM, Jhirad et al. (2023); Yin and Skiena (2023) used large language models such as GPT-3.5 and GPT-4 to discover DM with in-context learning tailored to diverse domains.

3 LM-LEXICON

Task Formulation. Given a set of golden references of definition d, the target of DM is to maximize the log-likelihood $\mathcal{P}$ of the generated hypothesis $\hat{d}$, which is conditioned on the input concatenated by a term of words t and a context c from the training set $\mathcal{D}$. Hence, we have $(c, t, d) \sim \mathcal{D}$, where the training set $\mathcal{D}$ contains a series of example triplets in the $<c, t, d>$ form. A concatenated sequence will be formatted with the given prompt template $p(\cdot, \cdot)$ as input to adapt the instruction-tuned LMs. For LMs with ICL, given a set of in-context pair-wise exemplars $\mathcal{E} \coloneqq \{e_i \triangleq (p(c_i, t_i) \oplus d_i) \mid i \in I\} \subseteq \mathcal{D}, s.t. \quad \forall i, j \in I, (c_i \neq c_j) \wedge (t_i \neq t_j) \wedge (d_i \neq d_j)$, we define

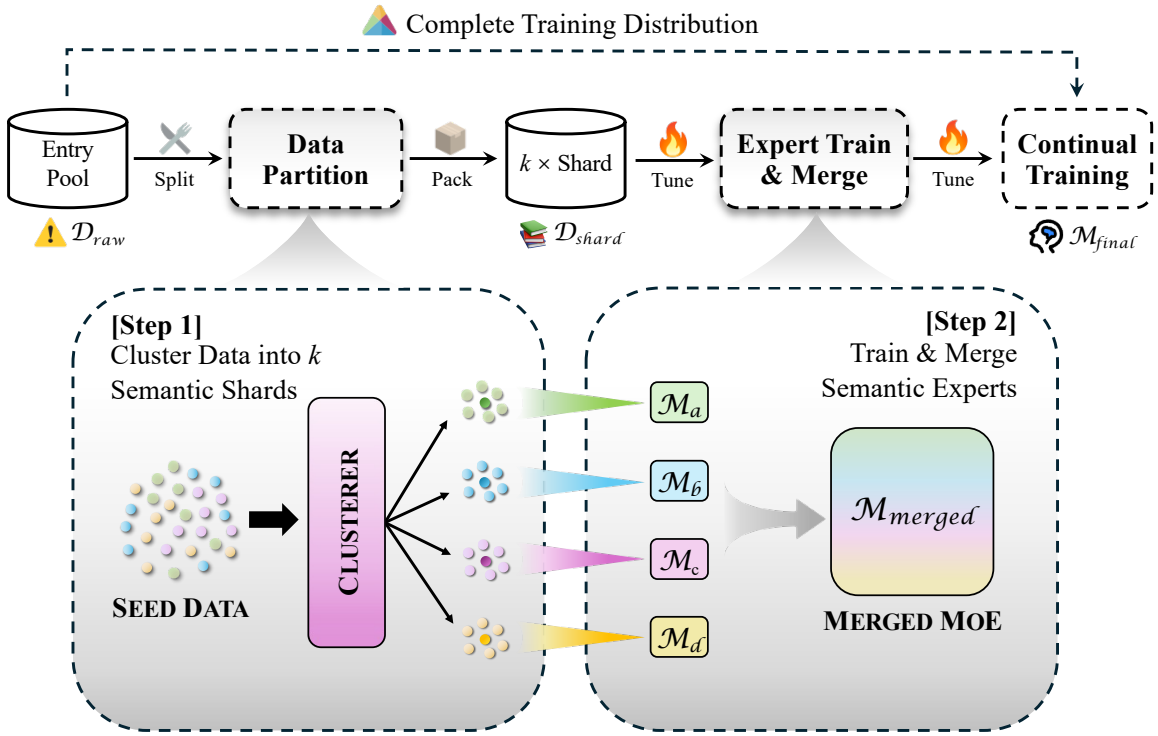

Figure 2: Diagram of LM-LEXICON training pipeline.

the input $\{e_i \oplus p(c_j, t_j) \mid e_i \in \mathcal{E}, (p(c_j, t_j) \oplus d_j) \notin \mathcal{E}\}, s.t. \forall i \neq j$ that combine a set of demonstration exemplars $\mathcal{E}$ to a templated prompt to aid generation across context[1]. Formally, for models with in-context learning, the causal generation procedure of each d is computed autoregressively using the joint conditional likelihood (Eq. 1).

$$\mathcal{P}(\hat{d} \mid \mathcal{E}, p(c,t)) = \prod_{i=1}^{|\hat{d}|} \mathcal{P}\left(\hat{d}_i \mid \hat{d}_{0:i-1}\right) \quad (1)$$

LM-LEXICON fine-tuned on $\mathcal{D}$ are optimized by the negative log-likelihood loss deriving from Eq. 1, which serves as causal language modeling objective that represents the average log-probability of a token, conditioned on previous tokens.

$$\mathcal{L}_{\text{DM}} = -\mathbb{E}_{(p,c,t)\sim\mathcal{D}}\left[\log \mathcal{P}(\hat{d} \mid p(c,t)) \cdot \mathbb{1}\right] \quad (2)$$

where $\mathbb{1}(x_i \in \hat{d})$ is an indicator function that is 1 if x_i from $\hat{d}$ is not a prompt token and 0 otherwise.

Data Clustering. Commonly, two distinct strategies are proposed for clustering on data: 1) lexical-based and 2) semantic-based methods. The lexical approach encompasses a straightforward, yet illustrative method: term frequency-inverse document frequency (TF-IDF) (Sparck Jones, 1972). In contrast, the semantic approach involves computing embeddings to represent the semantic meaning of a sequence (Zhang et al., 2019). As described in Equation 3, we follow the semantic-based paradigm to use balanced k-means (Malinen and Fränti, 2014) to construct our semantic-specialized experts via n-centroid clustering.

$$\min_{\{c_1,c_2,\ldots,c_K\}} \sum_{i=1}^{N}\sum_{k=1}^{K} \mathbb{1}(z_i = k) \cdot \|f(x_i) - c_k\|^2 \quad (3)$$

where $f(\cdot)$ denotes the selected embedder, z_i denotes the assigned cluster of data point x_i, and c_i represents the centroid of k-th cluster. $\mathbb{1}(z_i = k)$ is an indicator that equals 1 if the data point x_i is assigned to cluster k with index z_i, and 0 otherwise.

Model Architecture. For a deep transformer decoder, $\mathcal{M}$ (Vaswani et al., 2017), let $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbb{R}^{n\times d}$ denote the hidden states of input sequence in a transformer layer, where n denotes the number of tokens in a sequence, and d is the hidden dimension. LM-LEXICON consists of k experts, each of them has a multi-head attention (MHA) and a multi-layer perception (MLP) module in layer ℓ_i, where $\ell_i \in \mathcal{L} := \{\ell_0, \ell_1, \ldots, \ell_n\}$.

An MHA module divides the hidden state vectors into n_h different heads with a head dimension $d_h = d \,/\, n_h$. For each head h from a layer ℓ, differ-

[1] $\oplus$ indicates concatenating two strings with a blank.

ent weight matrices $W^{\ell}_{h,q}, W^{\ell}_{h,k}, W^{\ell}_{h,v} \in \mathbb{R}^{d_h \times d_h}$ are used to project the inputs into queries Q^{ℓ}_h, keys K^{ℓ}_h, and values V^{ℓ}_h. The attention scores and outputs in layer ℓ for the i-th token are computed as:

$$a^h_{ij} = \frac{\exp(\mathbf{q}^h_i {\mathbf{k}^h_j}^\top / \sqrt{d_h})}{\sum_{t=1}^{i} \exp(\mathbf{q}^h_i {\mathbf{k}^h_t}^\top / \sqrt{d_h})}, \ \mathbf{o}^h_i = \sum_{j=1}^{i} a^h_{ij} \mathbf{v}^h_j. \tag{4}$$

The outputs of all heads from layer ℓ_i are concatenated and linearly transformed to produce the final output $O_{\text{mha}} \in \mathbb{R}^{n \times d}$ from each expert.

$$O_{\text{mha}} = W_o[\mathbf{o}^1_i; \mathbf{o}^2_i; \cdots; \mathbf{o}^{n_h}_i] \tag{5}$$

where $W_o \in \mathbb{R}^{d \times d_h n_h}$, $i \in \{1, 2, \cdots, n\}$, and ; denotes the vector concatenation.

Let $X \coloneqq O_{\text{mha}}$ be the MLP inputs, the outputs of each expert e_i are routed by the gating layer $\mathcal{G}_i$. We then compute MLP output O_{mlp} as follows:

$$O_{\text{mlp}} = \sum_{i=1}^{n} \text{Softmax}(\text{TopK}(X \cdot W_{\mathcal{G}}))_i \cdot X \tag{6}$$

where TopK is the routing function that decides the activation of k selected experts for each token.

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

Input: Domain Experts $\mathcal{E} := \{e_1, e_2, e_3, e_4\}$.
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
    $i \leftarrow$ GetExpertIdx(e_i)
    /* Retrieve MHA and MLP weights */
    $\theta_{mha}, \theta_{mlp} \leftarrow$ HookWeights(e_i)
    for $\theta \in \{\theta_{mha}, \theta_{mlp}\}$ **do**
      if IsRouterLayer(θ) **then**
        /* Get formatted layer name */
        $n \leftarrow$ FormatName(θ, i)
        $\mathcal{M}[n] \leftarrow \theta$
      else ▷ AVERAGE θ OF MODULE
        $\mathcal{M}[n] \leftarrow \mathcal{M}$.get($n, \mathbf{0}$) $+ \theta / |\mathcal{E}|$
  return $\mathcal{M}$

Model Merging. Under the abstraction above, the MHA and MLP modules are composed of an expert to gather the parametric domain capability, as detailed in algorithm 1. Experts from $\mathcal{E}$ represent parallelly tuned seed models for data partitions. In the composition procedure, we first initialize a state dictionary for ℓ_i; subsequently, we iterate each expert and retrieve the MHA and MLP modules. We assign it as weights directly if the current weight serves as part of the router; otherwise, we average θ of modules, such as the embedding layer and LM head, and sum it up with the previous weights of modules of the same type.

To obtain semantic-aware experts after merging, we continue to train the gate layer $\mathcal{G}_i$ and experts to coordinate them in the same semantic representation space. See the details in Appendix §A.

4 Experimental Setup

Datasets. We now describe the datasets (see Table 1) that we used to train LM-LEXICON. We use the benchmarks introduced in Ishiwatari et al. (2019), which consist of four datasets and 3D-EX from Almeman et al. (2023) (see details in §A).

- **WordNet** (Noraset et al., 2017) is an online dataset[2] of terms, definitions, and examples.
- **Oxford** (Gadetsky et al., 2018) is built on the widely used online oxford dictionary[3].
- **Wikipedia**[4] (Ishiwatari et al., 2019) is introduced to focus testing the model capacity on phrases description, instead of words.
- **Urban** (Ni and Wang, 2017)[5] majorly contains terms of internet slang and urban words.
- **3D-EX** (Almeman et al., 2023) is the largest English definition modeling dataset[6] which includes plenty of well-known DM resources.

For $p(c, t)$ in Eq. 1 and 2, we follow Giulianelli et al. (2023) to use $p \coloneqq$ <BOS>"{{c}}" WHAT IS THE DEFINITION OF "{{t}}"<EOS> as the prompt template. We perform clustering only for 3D-EX and used the other four datasets as the natural clusters for our semantic experts training and merging.

Evaluation Metrics. To test performance of our methods and baselines, we employ six similarity-based metrics to evaluate the minimized expectation of difference between prediction $\hat{d}$ and reference d: $\min_{\pi_\theta} \mathbb{E}_{(c_i,t_i,d_i)\sim\mathcal{D}}[\log \mathbb{P}_{\pi_\theta}[\hat{d_i} \mid p(c_i, t_i)] - \log \mathbb{P}_{ref}[d_i \mid p(c_i, t_i)]]$, where π_θ is the weight of

[2] https://wordnet.princeton.edu
[3] https://en.oxforddictionaries.com
[4] https://www.wikidata.org
[5] https://www.urbandictionary.com
[6] 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".

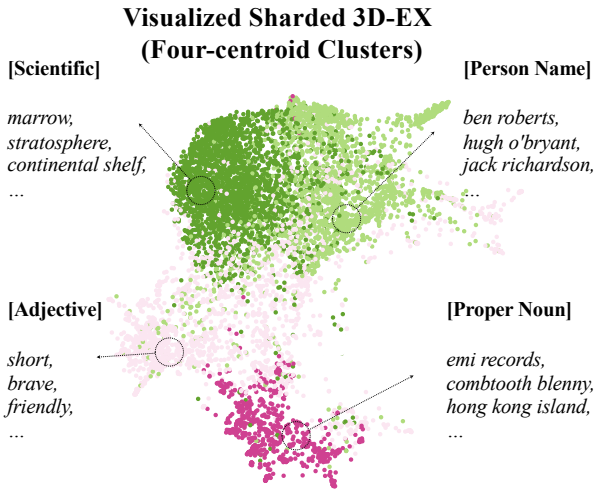

Figure 3: Four-cluster UMAP plot of 10K random definitions of terms in 3D-EX (§4). Each cluster is assigned manually with a **[label]** by their major constituents.

model $\mathcal{M}$ and triplet $(c_i, t_i, d_i) \in \mathcal{D}$. Specifically, at the lexical level, we used three generation metrics: BLEU (BL) (Papineni et al., 2002), ROUGE-L (RL) (Lin, 2004), and METEOR (MT) (Lavie and Agarwal, 2007). At the semantic level, we use BERTSCORE (BS) (Zhang et al., 2019), MOVERSCORE (MS) (Zhao et al., 2019), and MAUVE (MA) (Pillutla et al., 2021).

Compared Baselines. We select three types of strong baseline methods for comparison purposes.

- **Supervised Seq2seq LM:** We first replicate strong baselines finetuned on T5-base models (Raffel et al., 2020), including Reranked-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-Definition (Giulianelli et al., 2023).

- **Supervised Causal LM:** We report the in-distribution results of LlamaDictionary (Periti et al., 2024) and assess the out-of-distribution performance for the unseen datasets.

- **Frontier Causal LM:** We test LLM including GPT-4-Turbo (Achiam et al., 2023), Gemini-1.5-Pro (Reid et al., 2024), and Claude-3-Opus (Anthropic, 2024) with specific ICL strategy.

To compare with these baselines broadly, we replicate the setups used by prior work and reuse their reported results whenever possible.

5 Results and Analysis

5.1 Main Results

Following the training settings in §A, our major experimental results show that LM-LEXICON performs well in various metrics based on golden references (Table 2). The overall results show LM-LEXICON outperform significantly compared with the many-shot frotier models and strong supervised methods. Our best model (LM-LEXICON-MOE achieves the highest performance in Wordnet, Wiki,

[7] We observe and heuristically develop ad-hoc model-adapted parsers for prompt-based methods (proprietary models & ours) to extract our focused part of the generation.

	WordNet		Oxford		Wiki		Slang		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	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*	$36.99^{*}_{0.59}$	$37.83^{*}_{0.45}$	$26.09_{0.60}$	$34.55^{*}_{0.57}$	$57.9^{*}_{2.44}$	$\mathbf{59.56}^{*}_{1.50}$	$26.09^{*}_{0.27}$	$28.35^{*}_{0.28}$	$35.01^{*}_{0.22}$	$43.32^{*}_{0.27}$	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*	$37.73^{*}_{0.26}$	$37.94^{*}_{0.38}$	$26.74^{*}_{0.18}$	$35.18^{*}_{0.59}$	$59.33^{*}_{0.12}$	$59.46^{*}_{0.37}$	$26.73^{*}_{0.29}$	$28.54^{*}_{0.46}$	$34.83^{*}_{0.20}$	$42.68^{*}_{0.13}$	37.07* / 40.76*
LM-LEXICON-MOE (4×8B)											
↪ + *Zero-shot*	$40.09^{*}_{0.12}$	$40.51^{*}_{0.28}$	$23.35_{0.25}$	$32.94^{*}_{0.49}$	$60.31^{*}_{0.55}$	$55.52_{0.33}$	$\mathbf{31.26}^{*}_{0.85}$	$\mathbf{33.81}^{*}_{2.26}$	$45.38^{*}_{1.25}$	$45.21^{*}_{1.06}$	40.06* / 41.09*
↪ + *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*	$\mathbf{40.33}^{*}_{0.18}$	$40.69^{*}_{0.26}$	$24.18_{0.37}$	$33.79^{*}_{0.64}$	$\mathbf{60.88}^{*}_{0.55}$	$57.66_{0.73}$	$31.08^{*}_{0.17}$	$33.26^{*}_{0.22}$	$\mathbf{45.46}^{*}_{0.38}$	$\mathbf{45.73}^{*}_{0.26}$	**40.38* / 42.22***

Table 2: Main results on five benchmarks[7]. 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.

Slang, and 3D-EX, and achives the best average performance across all five benchmarks. Moreover, to consider the compute and performance gains tradeoff, we discuss the computational cost of our methods versus compared baselines in §B.

5.2 Additional Evaluation

LM-LEXICON Outperforms Larger Model A comparison of LM-LEXICON to other models in Figure 4 demonstrates that our method surpasses significantly larger dense models supervised by general instruction tuning, such as LLAMA-3.3-70B-INSTRUCT and LLAMA-3.1-405B-INSTRUCT, by +30.21% and +26.13% in BLEU, as well as +2.1% and +1.7% in BERT SCORE, respectively. In practice, we test two large dense models using the best 32-shot ICL settings. These results suggest merely scaling the model size may not enhance performance effectively in DM, and the efficacy of model specialization with our semantic-oriented sparsifying upcycling method.

Impact of Test-time Compute In light of Stiennon et al. (2020); Cobbe et al. (2021), we are curious on how to spur LM-LEXICON to achieve higher performance via test-time scaling, notably ground truth-based (Oracle) and Best-of-N (BoN) sampling with a outcome verifier (ORM). For oracle verifier, it uses reference as verification to provide binary feekbacks. When the verifier operates

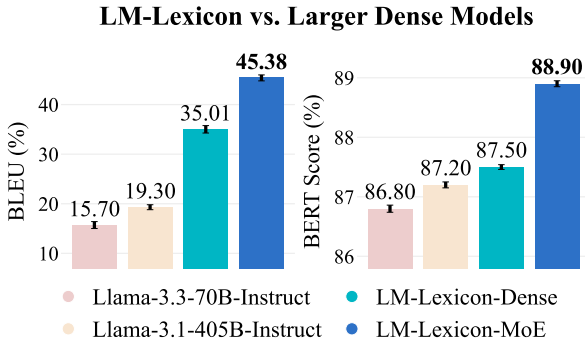

Figure 4: Test performance of BLEU on 3D-EX (LM-LEXICON versus larger dense models). We assess LLAMA-3.x models and report their 32-shot results.

as an ORM, it employs scalar feedback to select the optimal generation from candidates. As depicted in Table 2 (BoN-ORM), interestingly, it is observed that 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 that this is because of the diversity dimishment of models, as illustrated in (Brown et al., 2024). Intuitively, optimal results are achived via oracle verifier (Fig. 5) through repeated sampling with 128 completions per test case. Collaborating with the ORM or Oracle verifiers, LM-LEXICON's generation quality shows consistent improvements across the five benchmarks with the increase in the number of

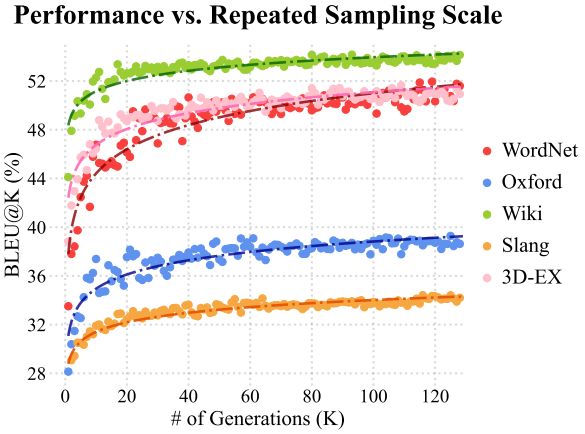

Figure 5: Repeated sampling results (BLEU) on five benchmarks evaluated by **oracle verifier**.

generations. This outcome aligns with the similar findings in mathematical reasoning tasks (Cobbe et al., 2021; Brown et al., 2024).

Scaling In-Context Learning To explore the ability and limitations of LLMs in definition modeling, we evaluated three strong models on WordNet performed across zero-, few-, and many-shot ICL.

As depicted in Figure 6, lexical similarities, including BLEU and ROUGE-L, improve with an increasing number of shots. Moreover, the semantic similarities among BERTSCORE, MOVERSCORE, and MAUVE exhibit the same trend. However, both GPT-4 and CLAUDE-3 begin to struggle with following task instructions when more <input, output> pairs are fed. In a particular range ($k \leq 32$), task performance improves as the exemplar number increases. However, it turns to degrade when $k > 32$, which indicates **too many exemplars hurt performance, even causing model collapse owing to increasing in-context semantic contradictions.** Hence, the optimization of DM with scaled ICL is limited. This finding is contrary to recent wisdom, which claims that many-shot prompting can continuously improve the model performance on many tasks (Agarwal et al., 2024; Bertsch et al., 2024; Li et al., 2024b).

5.3 Ablation Studies

We conduct ablation to study the crucial factors used in LM-LEXICON. In particular, we controlled for three keys: 1) the data partition method, 2) the routing policy, and 3) the number of experts.

Data Partition. 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: (i) no split, (ii) random split, and (iii) 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.

As plotted in Table 3, we observed that the original setting with semantic embedding clustering outperforms with about $+7\%$ gains in BLEU and $+1\%$ gains in ROUGE compared with lexical-based partition. 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 helps LM-LEXICON specialise robuster experts for each domain.

Model	BLEU	ROUGE	p-value
LM-LEXICON	$45.38_{\pm 0.3}$	$45.21_{\pm 0.1}$	–
+ w/o 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 results with the one-sided p-value in WordNet of various settings. The results are averaged over three runs.

Routing Policy. To obtain an in-depth comprehension on the routing policies and seek more effective semantic routing mechanism. Other than top-2 token-level routing, we experiment on (i) top-1 token-level; (ii) sequence-level; and (iii) domain-level. For token-level routing, we followed the implementation of (Fedus et al., 2022) and (Jiang et al., 2024). For sequence-level routing, we follow Pham et al. (2023). For domain-level routing, we implemented it inspired by Fan et al. (2021) and Gururangan et al. (2022). Table 4 presents

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

Table 4: Ablation on different routing policies. Results are averaged over three runs. We provide a one-sided p-value to inspect whether they satisfy t-test assumptions.

that the domain-level routing are the most effcient, even surpassing the top-2 token-level routing (LM-LEXICON), indicating semantic-routing by specified domain cluster may be more beneficial for

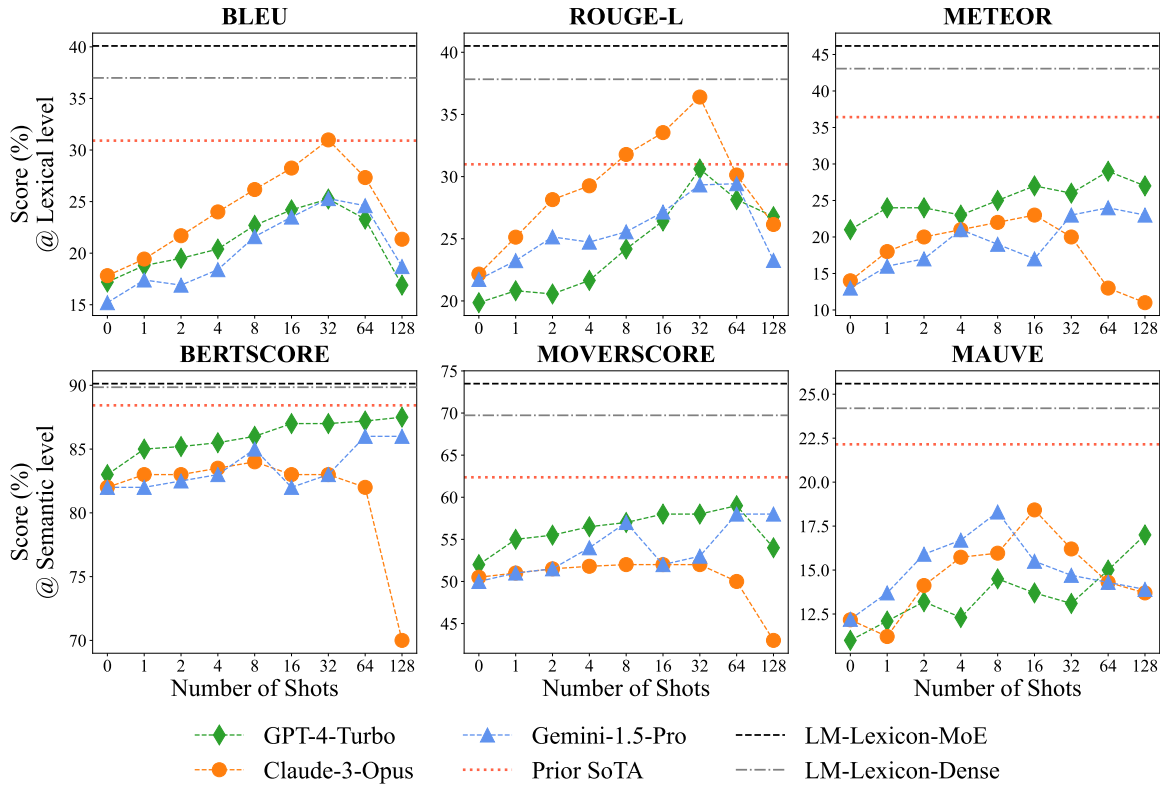

Figure 6: Scaling the in-context learning results of frontier causal LMs on WordNet with k-shot demonstrations, where k scales logarithmically from 0 to 128. Prior SoTA denotes the Rerank-T5 proposed by Huang et al. (2021).

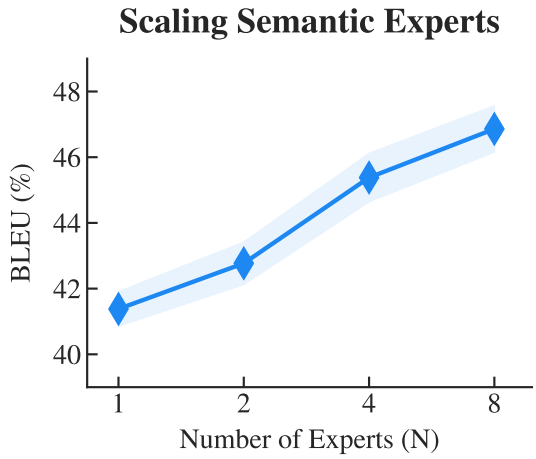

Figure 7: Scaling test performance of (measuring with BLEU) on 3D-EX, with different number of experts.

semantic-intensive tasks.

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

As shown in Figure 7, we find that across all settings of N, our model 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 effectiveness and scalability of our method utilizing more semantic experts. This increasing trend can potentially be extended by integrating more fine-grained semantic experts, as illustrated by Dai et al. (2024) and He (2024), but we leave this direction for future work.

6 Conclusion

In this paper, we present LM-LEXICON, an approach that explores, specializes, and combines domain experts upcycling to a sparse MoE model, which can generate appropriate definitions of terms for various domains and genres. Empirically, we show that LM-LEXICON significantly outperform current frontier LLMs and a series of strong supervised baselines by data clustering and sparse upcycling. We hope LM-LEXICON could be promisingly extended to more domains and semantic-intensive tasks in the future.

Limitations

Stronger Verifier. Our results from Section 5 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*.

Rishabh Agarwal, Avi Singh, Lei M Zhang, Bernd Bohnet, Stephanie Chan, Ankesh Anand, Zaheer Abbas, Azade Nova, John D Co-Reyes, Eric Chu, and 1 others. 2024. Many-shot in-context learning. *arXiv preprint arXiv:2404.11018*.

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*.

Amanda Bertsch, Maor Ivgi, Uri Alon, Jonathan Berant, Matthew R Gormley, and Graham Neubig. 2024. In-context learning with long-context models: An in-depth exploration. *arXiv preprint arXiv:2405.00200*.

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*.

Maximillian Chen, Xiao Yu, Weiyan Shi, Urvi Awasthi, and Zhou Yu. 2023. Controllable mixed-initiative dialogue generation through prompting. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 951–966.

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.

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*.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav

Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Michael Auli, and Armand Joulin. 2021. Beyond english-centric multilingual machine translation. *Journal of Machine Learning Research*, 22(107):1–48.

Mariia Fedorova, Andrey Kutuzov, and Yves Scherrer. 2024. Definition generation for lexical semantic change detection. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 5712–5724, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

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.

Besnik Fetahu, Zhiyu Chen, Oleg Rokhlenko, and Shervin Malmasi. 2023. InstructPTS: Instruction-tuning LLMs for product title summarization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 663–674, Singapore. Association for Computational Linguistics.

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, Mike Lewis, Ari Holtzman, Noah A. Smith, and Luke Zettlemoyer. 2022. DEMix layers: Disentangling domains for modular language modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5557–5576, Seattle, United States. 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*.

Xu Owen He. 2024. Mixture of a million experts. *arXiv preprint arXiv:2407.04153*.

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2015. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics,Transactions of the Association for Computational Linguistics*.

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.

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.

Jiahuan Li, Hao Zhou, Shujian Huang, Shanbo Cheng, and Jiajun Chen. 2024a. Eliciting the translation ability of large language models via multilingual finetuning with translation instructions. *Transactions of the Association for Computational Linguistics*, 12:576–592.

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*.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2024b. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*.

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.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, The Google Books Team, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182.

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.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Xingyuan Pan, Luyang Huang, Liyan Kang, Zhicheng Liu, Yu Lu, and Shanbo Cheng. 2024. G-dig: Towards gradient-based diverse and high-quality instruction data selection for machine translation. *arXiv preprint arXiv:2405.12915*.

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. *arXiv preprint arXiv:2403.04894*.

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.

Xiao Pu, Mingqi Gao, and Xiaojun Wan. 2023. Summarization is (almost) dead. *arXiv preprint arXiv:2309.09558*.

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

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. *Journal of machine learning research*, 21(140):1–67.

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*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, and 1 others. 2021. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

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*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

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.

Zihao Yi, Jiarui Ouyang, Yuwen Liu, Tianhao Liao, Zhe Xu, and Ying Shen. 2024. A survey on recent advances in llm-based multi-turn dialogue systems. *arXiv preprint arXiv:2402.18013*.

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*.

Yusen Zhang, Nan Zhang, Yixin Liu, Alexander Fabbri, Junru Liu, Ryo Kamoi, Xiaoxin Lu, Caiming Xiong, Jieyu Zhao, Dragomir Radev, Kathleen McKeown, and Rui Zhang. 2024. Fair abstractive summarization of diverse perspectives. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3404–3426, Mexico City, Mexico. Association for Computational Linguistics.

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.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. 2024. LMSYS-chat-1m: A large-scale real-world LLM conversation dataset. In *The Twelfth International Conference on Learning Representations*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, and 1 others. 2024. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36.

Tong Zhu, Xiaoye Qu, Daize Dong, Jiacheng Ruan, Jingqi Tong, Conghui He, and Yu Cheng. 2024a. 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.

Wenhao Zhu, Hongyi Liu, Qingxiu Dong, Jingjing Xu, Shujian Huang, Lingpeng Kong, Jiajun Chen, and Lei Li. 2024b. Multilingual machine translation with large language models: Empirical results and analysis. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2765–2781, Mexico City, Mexico. 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. 8) 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.

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.

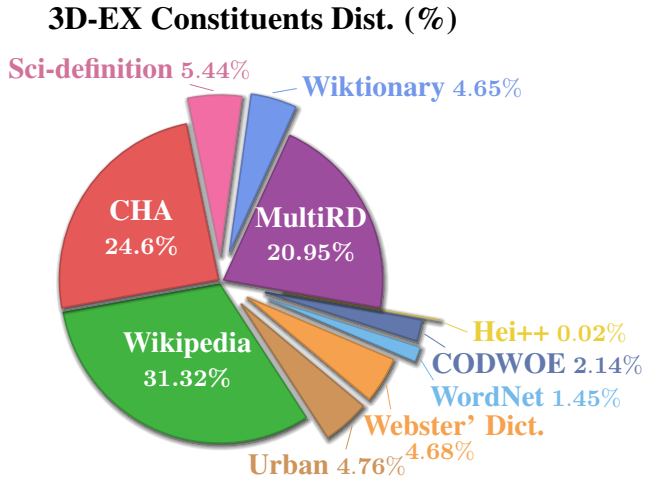

Figure 8: 3D-EX constituents distribution.

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.

Cluster Settings 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 §5.3.

Training Details LM-LEXICON was trained for 3 epochs with a global batch size of 4,096 tokens (gradient accumulation 1, batch size per device 8, max sequence length 128) on 4 × 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. 8, We set up the following hyperparameters 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. 2), 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. 7) to frame the reward modeling as a binary classification objective.

$$\mathcal{L}_{\text{ORM}} = -\log \sigma \left(r_\phi(x, y_w) - r_\phi(x, y_l) \right) \quad (7)$$

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 Settings 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 evlaluation, we follow (Huang et al., 2021) and refactor their implementation with concurrent metrics com-

putation to boost the inference procedure in large models, please see the details in our released code.

B Carbon Footprint

The cost of fine-tuning large language models 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 of our experiments. Each experiment was conducted using our own infrastructure. We calculated the carbon footprint estimation using a carbon intensity of 0.141 kg/kWh and 700 W consumption per GPU[8].

Model	Hardware	FLOPs	Time (h)	CO2eq (kg)
M-LEXICON-DENSE	8×H100	$4.2e^{18}$	36.4	11.4
M-LEXICON-MOE	8×H100	$5.4e^{18}$	32.8	14.6

Table 6: Details about the training required resources.

C Additional Evaluation Results

As depicted in Figure 9, 10, 11, and 12, we provide a cherry-picked example for each domain cluster as shown in Figure 3 in definition modeling.

Cluster-1 Example:
[Term] Combtooth Blenny
[Query] "the crested blenny is a species of **Combtooth Blenny** found around New South Wales, Australia, and New Zealand to depths of between ..." What is the definition of "**Combtooth Blenny**"?
[Source] Wikipedia
[Reference] **Combtooth Blenny**: perciform marine fish of the family blenniidae.

Figure 9: 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 face and deal with danger or fear without flinching.

Figure 10: Example of $\mathcal{C}_2$ (adjective) from 3D-EX.

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

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 11: 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 disease and late-onset hereditary lymphedema are sometimes associated with distichiasis ..." 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 (part of the immune system that produces and transports fluids and immune cells throughout the body).

Figure 12: Example of $\mathcal{C}_4$ (scentific) from 3D-EX.

D Code for

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 7.

Algorithm 2 Pytorch code for semantic experts merger.

```
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 ...
```

```
    # 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
```

	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 7: 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.

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 8: 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 🤗 ugging Face Transformers (Wolf et al., 2020).