Gold Panning in Vocabulary: An Adaptive Method for Vocabulary Expansion of Domain-Specific LLMs

Chengyuan Liu[1,2]*, Shihang Wang[2], Lizhi Qing[2], Kun Kuang[1]†, Yangyang Kang[3,1,2], Changlong Sun[2], Fei Wu[1]
{liucy1,kunkuang,yangyangkang,wufei}@zju.edu.cn,
{wangshihang.wsh,yekai.qlz}@alibaba-inc.com, changlong.scl@taobao.com
[1]College of Computer Science and Technology, Zhejiang University,
[2]Tongyi Lab, Alibaba Group,
[3]Polytechnic Institute, Zhejiang University

Abstract

While Large Language Models (LLMs) demonstrate impressive generation abilities, they frequently struggle when it comes to specialized domains due to their limited domain-specific knowledge. Studies on domain-specific LLMs resort to expanding the vocabulary before fine-tuning on domain-specific corpus, aiming to decrease the sequence length and enhance efficiency during decoding, **without thoroughly investigating the results of vocabulary expansion to LLMs over different domains**. Our pilot study reveals that expansion with only a subset of the entire vocabulary may lead to superior performance. **Guided by the discovery, this paper explores how to identify a vocabulary subset to achieve the optimal results.** We introduce VEGAD, an adaptive method that automatically identifies valuable words from a given domain vocabulary. Our method has been validated through experiments on three Chinese datasets, demonstrating its effectiveness. Additionally, we have undertaken comprehensive analyses of the method. The selection of a optimal subset for expansion has shown to enhance performance on both domain-specific tasks and general tasks, showcasing the potential of VEGAD.

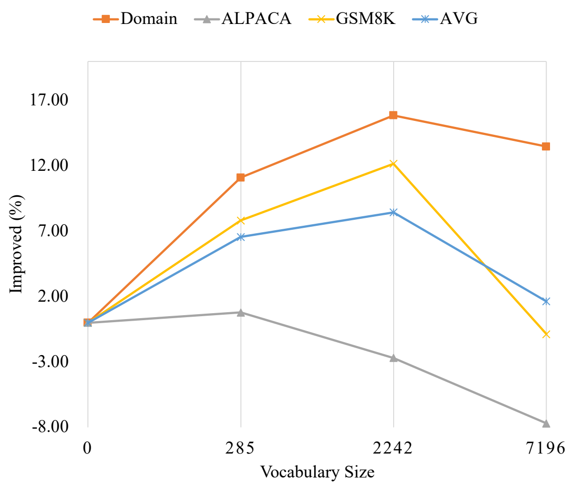

Figure 1: Pilot study: Relative improvement comparing with direct supervised fine-tuning, by adding vocabulary with different sizes.

1 Introduction

Despite achieving satisfactory performance on a wide range of tasks (OpenAI et al., 2024; Touvron et al., 2023a; Xu et al., 2023; Yuan and Zhu, 2023), Large Language Models (LLMs) continue to encounter challenges, particularly in domain-specific tasks, such as the generation of legal, medical, and financial texts. The expansion of vocabulary (Provilkov et al., 2020; Liu et al., 2021; Ozdemir and Goksel, 2019; Rothe et al., 2020) serves as a strategy to enhance the decoding efficiency for domain-specific LLMs. By concatenating specific, frequent n-grams into new words, the token sequence is shortened, thereby visibly boosting efficiency. Cui et al. (2024) extended LLaMA's existing vocabulary with an additional 20,000 Chinese tokens, thereby improving its encoding efficiency and semantic understanding of Chinese. LawGPT[1] is fine-tuned based on the general Chinese LLMs (such as Chinese-LLaMa, ChatGLM (Du et al., 2022), etc.), the legal domain specific vocabulary is expanded to enhance the semantic understanding ability of the LLMs.

Current researches primarily focus on some specific domains. Nonetheless, they have not thoroughly elucidate the performance enhancements resulting from vocabulary expansion in various domains. We conduct a pilot study illustrating the domain performance and general capabilities after vocabulary expansion with different sizes, and the results are illustrated in Figure 1. It is revealed that **augmenting the size of the newly added vo-**

*This work was done when Chengyuan Liu interned at Alibaba.
†Corresponding author.

[1]https://github.com/pengxiao-song/LaWGPT

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 7442–7459
November 12-16, 2024 ©2024 Association for Computational Linguistics

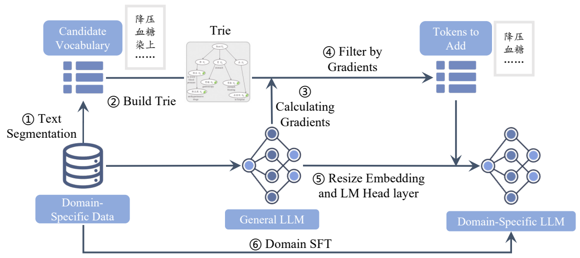

Figure 2: Framework of VEGAD.

cabulary does not invariably result in improved model performance. Hence, an essential question arises regarding **the generation of an optimal subset for vocabulary expansion given a candidate vocabulary.** The process of selecting high-value vocabulary during the expansion of domain-specific LLMs is akin to *gold panning*, as it requires careful selection rather than indiscriminate enlargement of the lexicon to enhance the performance of the LLMs. We recognize the following challenges for vocabulary subset generation:

- *How to ensure an optimal performance over the whole vocabulary?*
- *How to automatically adapt to any domain?*

To effectively identify the crutial words from a candidate vocabulary, we have proposed VEGAD (abbreviation of "**V**ocabulary **E**xpansion via **G**r**AD**ients"), which is an adaptable vocabulary expansion method via gradients. Figure 2 provides an illustration of the framework. Intuitively, token groups displaying larger gradients in domain instances are deemed more pivotal to the task and should be integrated into the vocabulary as domain-specific terms. Therefore, it is a straightforward approach to trace the gradient of each word, while there are several difficulties, such as the algorithm to efficiently retrieve the candidate words from the token sequences, and the gradient calculation across various tokens rather than the whole sequence. To identify candidate words from the token sequences, we build a Trie (Black, 2019) based on the candidate vocabulary, and design an algorithm to record the gradient for each word with the Trie. To distinguish the effect of each token, the gradient is calculated on the running tensors, instead of the weights of the LLMs.

To scrutinize the efficacy of VEGAD, we have undertaken comprehensive studies. The findings across three Chinese datasets, pertaining to the domains of law and medicine, underscore a superiority in comparison to other lexicon generation techniques, as well as the promising prospects of domain-specific vocabulary expansion. Our inquiry reveals that the domain-specific lexicon by VEGAD enhances performance in tasks requiring specialized knowledge as well as tasks demanding general skills. We hope that our multi-perspective analysis serves as a catalyst for future investigations into enhancing domain-task performance and mitigating the Catastrophic Forgetting through domain vocabulary adaptation.

In summary, our contributions are three folds:

- It is revealed by our pilot study that vocabulary expansion with only a subset of the entire supplementary domain vocabulary may lead to superior performance over using the whole vocabulary.
- Guided by our discovery, we introduce VEGAD, an automatic method to effectively identify an optimal subset for vocabulary expansion, adaptable to various domains.
- Extensive experiments and analyses have been performed, during which VEGAD displays outstanding proficiency surpassing other vocabulary expansion methods.

2 Related Work

Large Language Models, such as ChatGPT[2], GPT-4 (OpenAI et al., 2024), exhibit amazing abilities on understanding and text generation. They can handle the tasks of QA, reasoning and math calculation even under zero-shot scenarios. LLaMa (Touvron et al., 2023a) is a collection of open foundation language models ranging from 7B to 65B parameters. Touvron et al. (2023b) developed and released Llama 2, a collection of LLMs ranging in scale from 7B to 70B parameters. The fine-tuned Llama 2-Chat, are optimized for dialogue use cases. There are other popular LLMs developed with various skills (Rozière et al., 2024; Almazrouei et al., 2023; Jiang et al., 2023; Bai et al., 2023; Baichuan, 2023).

Due to the lack of domain-specific knowledge, general LLMs fall short at handling domain questions. Therefore domain-specific LLMs are developed by fine-tuning on domain corpus. (Xiong

[2]https://chat.openai.com/

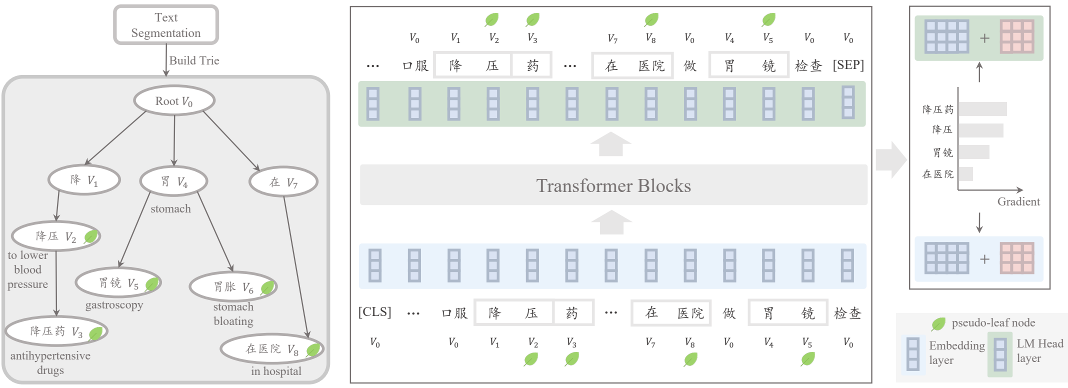

Figure 3: Gradient Calculation for each candidate word. Given the Trie built from candidate vocabulary, we check whether there exists a sub-sequence of the input and output on the path from the root of the Trie to a leaf node, by a pointer. The trace of the pointer is illustrated by V_i and the "pseudo-leaf node". Finally, the top K words with the largest gradients are selected to construct the new vocabulary, and used to resize the embedding layer and language modeling head layer.

et al., 2023) collected databases of medical dialogues with the help of ChatGPT and adopted several techniques to train an easy-deploy LLM, called DoctorGLM. Wang et al. (2023a) proposed HuaTuo, a LLaMA-based model that has been supervised-fine-tuned with generated QA (Question-Answer) instances in biomedical domain tasks, with medical expertise in the responses. Cui et al. (2023) proposed an open-source legal LLM named ChatLaw, with a method that combines vector database retrieval with keyword retrieval to effectively reduce the inaccuracy of relying solely on vector database retrieval, and a self-attention method to enhance the ability to overcome errors present in reference data. There are other domains studied including finance (Wang et al., 2023b; Yu, 2023), education (Yu et al., 2023a), science (Li et al., 2023b) and e-commerce (Li et al., 2023a).

Several previous studies adopt a strategy, vocabulary expansion, to improve the performance of domain SFT. Specifically, a domain-specific vocabulary is automatically generated or manually designed, and added into the tokenizer. In order to augment LLaMA with capabilities for understanding and generating Chinese text and its ability to follow instructions, Cui et al. (2024) extended LLaMA's existing vocabulary with an additional 20,000 Chinese tokens, thereby improving its encoding efficiency and semantic understanding of Chinese. Liu et al. (2023) proposed task-adaptive tokenization as a way to adapt the generation pipeline to the specifics of a downstream task and enhance long-form generation in mental health. However, their task-adaptive tokenizer samples variable segmentations from multiple outcomes, which may change the vanilla behaviour of other tokenizers (e.g., WordPiece and BPE). LaWGPT expands the legal domain specific vocabulary and large-scale Chinese legal corpus pre-training on the basis of the general Chinese base model (such as Chinese-LLama, ChatGLM, etc.), and enhances the basic semantic understanding ability of the LLM in the legal field. Tongyi-Finance-14B[3] expanded the vocabulary of financial domain in Qwen-14B, and the size of the vocabulary is 150,000. Based on the BPE vocabulary used in GPT-4, the vocabulary is optimized for Chinese and multi-language. The numbers are divided into individual digits. Liu et al. (2024b) identified tokens that are absent in the general-purpose tokenizer and are rarely found in general-purpose datasets, from the vocabulary of the new tokenizer. They initialize model embeddings of the new tokens by utilizing the general-purpose tokenizer. Liu et al. (2021) introduced two new approaches based on attention to initialize the weights of new added words.

3 Method

In this Section, we introduce VEGAD, a vocabulary expansion method via gradient for domain-specific LLMs. The process is shown in Figure 3.

Our approach is inspired by an naive intuition: n-gram tokens exhibiting larger gradients in response to domain-specific instances are deemed crucial for the task at hand, and therefore, warrant inclu-

[3]https://modelscope.cn/models/TongyiFinance/Tongyi-Finance-14B

Algorithm 1 Build Trie

Require: $\mathcal{W}_1, \mathcal{W}_2, \cdots, \mathcal{W}_n, n, V_0$

$root \leftarrow V_0$
$M \leftarrow 1$
for $i = 1 \rightarrow N$ **do**
  $p \leftarrow root$
  for $t_i^j \in \mathcal{W}_i$ **do**
    if p has child t_i^j **then**
      $p \leftarrow \textbf{GetChild}(p, t_i^j)$
    else
      $V_M \leftarrow \textbf{CreateChild}(p, t_i^j)$
      $p \leftarrow V_M$
      $M \leftarrow M + 1$
    end if
  end for
  set p as pseudo-leaf node
end for

sion in the lexicon as domain-specific terminology. Nonetheless, there are several challenges. For example, the algorithm to efficiently retrieve the candidate words from the token sequences, and the gradient calculation across various tokens rather than the whole sequence.

Specifically, starting from the domain-specific data, sentences are divided into discrete words. The candidate vocabulary is constructed with words absent from the general lexicon. Subsequently, the process of selection is executed on domain-specific instances by computing the gradients for each node within the embedding tensor and the language modeling tensor, with reference to a Trie constructed based on the candidate vocabulary. The top K words exhibiting the highest overall gradients are retained to establish the specialized domain vocabulary. Then we resize the LLM and incorporate the tokenizer with new vocabulary, following an optional weight initialization. Then we conduct domain SFT on the LLM, to develop the domain-specific LLM.

The advantage of VEGAD can be summarized as following: **1)** VEGAD is a plug-and-play task-adaptive vocabulary selection method, seamlessly integrating with diverse techniques utilized in supervised fine-tuning. **2)** In contrast to previous methods such as Liu et al. (2023), which might alter the intrinsic behaviors of current tokenizers such as WordPiece and BPE by imposing an obligatory scoring mechanism for sampling in accordance with their guidelines, VEGAD is tokenizer-agnostic, and compatible to any tokenization algorithms. **3)** The pipeline is automatically performed, without the need of manual design or intervention. Of course, it still allows additional edition to the vocabulary if required.

3.1 Build Trie

The Trie, as discussed by Black (2019), represents a distinct tree-based data structure, extensively employed within the realm of computer science for the administration of dynamic sets or associative arrays, with the keys predominantly being strings. Diverging from the structure of a binary search tree in which a node's placement is influenced by numerical or logical hierarchy, in a Trie, the location of a node is unequivocally defined by the sequence of characters it denotes. We illustrate an example of Trie in the left part of Figure 3.

Formally, the domain-specific dataset can be represented as $D = \{(X_1, Y_1), \cdots, (X_n, Y_n)\}$, where X and Y are the query and response respectively, n is the size of D. Given a text segmentation tool, the candidate vocabulary is constructed following

$$\mathcal{V} = (\bigcup_{i=1}^{n} \textbf{Segment}(X_i)) \cup (\bigcup_{i=1}^{n} \textbf{Segment}(Y_i)) \tag{1}$$

The candidate vocabulary is denoted as $\mathcal{V} = \{w_1, w_2, \cdots, w_N\}$, where N denotes the size of the candidate vocabulary. Then we build the Trie based on candidate vocabulary. For the i-th word w_i, we tokenize it to several tokens with the existing general tokenizer:

$$\mathcal{W}_i = \textbf{tokenize}(w_i) = [t_i^1, t_i^2, \cdots, t_i^{l_i}] \tag{2}$$

Note that $l_i > 1$ because each word in the candidate vocabulary doesn't exist in the general tokenizer's lexicon. Let V_0 be the root of the Trie. For each word w_i, we insert its tokens one by one into the Trie, starting from V_0. Additionally, we set a flag of "pseudo-leaf node" to each $t_i^{l_i}$ node, which is the last token of the word w_i[4]. Note that each path from the root to a "pseudo-leaf node" represents a candidate word in $\mathcal{V}$. The procedure is illustrated in Algorithm 1. With the algorithm, we get a Trie with M nodes.

[4]The "pseudo-leaf node" is different from the traditional concept of "leaf node" in tree-based data structures. There may be children nodes for "pseudo-leaf node", because some token sequence $\mathcal{W}_j$ may start from another $\mathcal{W}_i$.

3.2 Gradient Calculation

With the general tokenizer, the sentences are converted to input query tokens and output response tokens. For simplicity, the input and output sequence of the LLM are denoted as $x = [x_1, \cdots, x_L]$ and $y = [y_1, \cdots, y_L]$ respectively, where L is the length of the sequences. Current LLMs firstly embed the input tokens to α in a high-dimension space, then perform transformers on the embedding vectors α. The representation h output by several transformer blocks is finally converted to the distribution $\hat{y}$ over tokens through a language modeling head layer:

$$\alpha = \mathbf{Embed}(x) \tag{3}$$

$$h = \text{Transformers}(\alpha) \tag{4}$$

$$\hat{y} = h \times \mathbf{LMHead}^{\mathsf{T}} \tag{5}$$

where $\mathbf{Embed}, \mathbf{LMHead} \in \mathbb{R}^{C\times d}$, C and d denote the size of vanilla vocabulary and the dimension. The standard language modeling loss is adopted:

$$\begin{aligned}\mathcal{L}_{\text{lm}} =& -\sum_{i=1}^{L} \log p(y_i|x_{<i}) \\ =& \mathbf{CrossEntropy}(y, \text{Softmax}(\hat{y}))\end{aligned} \tag{6}$$

For the embedding tensor, we calculate the gradients of each input token as G^{embed}. Although previous studies mostly only focus on the embedding layer, we find that the language modeling head layer is also important especially for text generation tasks. Therefore, we calculate the gradients G^{lmhead} for each output token only if it is not a special token (e.g., `[CLS]`, `[SEP]` and `[PAD]`). To obtain the gradient at each time step, Equation 5 is modified as:

$$\hat{y} = \beta \otimes (h \times \mathbf{LMHead}^{\mathsf{T}}) \tag{7}$$

where $\beta \in \mathbb{R}^{L\times C}$ is filled with 1, and $\otimes$ denotes element-wise production.

$$G^{\text{embed}} = \frac{\partial \mathcal{L}_{\text{lm}}}{\partial \alpha}, G^{\text{lmhead}} = \frac{\partial \mathcal{L}_{\text{lm}}}{\partial \beta} \tag{8}$$

Then we calculate the gradient for each candidate word by looking up nodes in the Trie and iterating over x and y. The candidate words appearing in the sequence can be identified by moving a pointer from the root V_0 initially. During enumerating i from 1 to L, we check if there exists a sub-sequence $x_{i:j}$ in Trie. Specifically, from the root, the pointer constantly moves to its children until it reaches the last "pseudo-leaf node" or the token mismatches any child of the current node. Once the pointer reaches a node V' attributed with "pseudo-leaf node", we add the norm of the gradients of the sub-sequence to w, where w denotes the candidate word represented by V'.

$$\begin{aligned}G_w = G_w &+ ||\sum_{q=i}^{j} G_q^{\text{embed}}||_2 \\ &+ ||\sum_{q=i-1}^{j-1} G_q^{\text{lmhead}}||_1\end{aligned} \tag{9}$$

Note that there is a position shift for the output sequence (i.e. $x_{i:j} = y_{i-1:j-1}$). We provide the detailed code in Algorithm 2.

To enhance efficiency, the algorithm's cost of time can be optimized by adopting prefix accumulation in conjunction with the Aho–Corasick Algorithm. This optimization is particularly significant in cases involving Tries of considerable size and depth, resulting in a notable reduction in the algorithm's overall complexity. The detailed optimization is described in Appendix L.

3.3 Vocabulary Selection

Upon evaluating the gradient associated with each word from the candidate vocabulary, the words are organized in descending order based on the magnitude of their gradients. We obtain the top K words and remove other words. These selected words are then integrated into the pre-existing general vocabulary. The embedding layer and language modeling head layer are also resized to $\mathbb{R}^{(C+K)\times d}$.

For initialization, the default method is averaging the weights of sub-tokens in the original layer, following Liu et al. (2023). We also investigated other approaches and the results are discussed in Appendix G.

4 Experiments

The main results on three datasets from two domains are discussed in SubSection 4.2. Then we discuss the influence of the vocabulary size in SubSection 4.3. To verify our hypothesis, we compare the words with different gradients in Appendix C. We also remove the pre-built candidate vocabulary, to investigate the influence of direct gradient calculation on 2-gram tokens of the sequence in Appendix D. There are also discussions about

Method	Article QA				ALPACA		GSM8K			Safety Prompts
	BLEU	ROUGE-1/2/L			BLEU	ROUGE	ACC	BLEU	ROUGE	ACC
General LLM	10.28	29.50	10.00	20.93	11.57	23.55	22.10	21.33	33.63	94.00
SFT	26.70	46.53	24.53	36.60	12.19	25.15	14.40	19.17	31.55	88.30
DV	26.23	47.10	24.83	36.71	12.11	25.11	14.50	**19.86**	**32.14**	88.70
SPM	25.56	45.77	24.83	36.02	**12.56**	24.89	8.10	17.85	30.33	88.70
+ATT_EG	24.31	45.06	22.82	34.89	12.07	24.72	8.30	17.99	30.56	89.40
+PATT_EG	25.96	45.98	24.01	36.22	11.99	24.63	8.50	17.95	30.57	89.50
Jieba	28.04	48.36	26.88	38.25	11.97	24.64	6.60	18.15	30.63	88.30
VEGAD	**28.58**	**48.67**	**26.96**	**39.11**	12.39	**25.43**	**15.20**	19.85	**32.14**	**89.60**

Table 1: Results on Article QA of legal domain.

Method	Article QA	GSM8K		Safety Prompts	AVG
	BLEU	ACC	BLEU	ACC	-
SFT	+159.73	-34.84	-10.13	-6.06	+22.81
DV	+155.16	-34.39	**-6.89**	-5.64	+22.58
SPM	+148.64	-63.35	-16.32	-5.64	+14.38
+ATT_EG	+136.48	-62.44	-15.66	-4.89	+11.56
+PATT_EG	+152.53	-61.54	-15.85	-4.79	+14.80
Jieba	+172.76	-70.14	-14.91	-6.06	+17.02
VEGAD	**+178.02**	**-31.22**	-6.94	**-4.68**	**+28.45**

Table 2: Relative improvement after SFT on Article QA, comparing to general LLM. The metrics are reported in percentage.

the influence of the language modeling head layer, model scale and weight initialization methods in Appendix E, F and G, respectively.

Our study incorporates three domain-specific datasets from two distinct domains: Article QA dataset for the legal domain, and CMedQA (Zhang et al., 2018) and CMDD (Toyhom, 2023) datasets for the medical field. Furthermore, we delve into the Catastrophic Forgetting issue in general tasks following supervised fine-tuning on domain-specific instances. To this end, we analyze three datasets: ALPACA (Peng et al., 2023) for tasks requiring instruction following, GSM8K (Yu et al., 2023b) focused on mathematics, and SafetyPrompts (Sun et al., 2023) concerning safety. The metrics and details of the dataset consideration and construction are described in Appendix A.

4.1 Baselines

General LLM The LLM fine-tuned on general tasks. It is mainly considered as the reference when studying CF problem.

SFT Direct supervised fine-tuning on domain-specific dataset.

DV We adopt domain concepts and terminology as the vocabulary to be added. For legal domain, the expert-designed legal vocabulary by LawGPT[5] is used. For medical domain, we prompt GPT-4 to extract the names of medicine, symptom and therapies from the sentences. We keep words that appear more than 100 times in the data to improve the effectiveness, because increasing the size of the newly added vocabulary does not invariably result in improved model performance, according to our experiment in SubSection 4.3.

SPM We train a tokenizer with SentencePiece (Kudo and Richardson, 2018), which is a common method to generate domain-specific vocabulary (Cui et al., 2024). We utilize the off-the-shelf package[6].

ATT_EG and PATT_EG Liu et al. (2021) introduced two weight initialization methods based on attention mechanism, ATT_EG and PATT_EG. They apply the methods on the generated vocabulary by SPM for downstream tasks.

Jieba Inspired by SPM, we adopt another text segmentation tool, Jieba[7]. From the experiments, we find it to be a strong and convenient baseline for text generation tasks.

Implementation details are shown in Appendix B.

4.2 Main Results

4.2.1 Legal Domain

The outcomes for Article QA are presented in Table 1, and the relative improvements are shown in Table 2. **1)** Within the array of baseline comparisons, Jieba demonstrates superior performance in domain-specific tasks. Specifically, Jieba achieves a BLEU score that is 1.3 points greater than that

[5]https://github.com/pengxiao-song/LaWGPT/blob/main/resources/legal_vocab.txt
[6]https://github.com/google/sentencepiece
[7]https://github.com/fxsjy/jieba

Method	CMedQA				ALPACA		GSM8K			SafetyPrompts
	BLEU	ROUGE-1/2/L			BLEU	ROUGE	ACC	BLEU	ROUGE	ACC
General LLM	3.15	17.46	2.27	14.40	11.57	23.55	22.10	21.33	33.63	94.00
SFT	3.29	19.85	3.94	14.30	9.19	21.42	16.20	11.40	28.95	87.80
DV	3.61	19.24	3.88	14.32	9.61	22.01	**17.60**	11.67	29.56	88.50
SPM	3.29	18.91	3.61	13.88	9.15	21.34	8.60	12.13	28.29	85.20
+ATT_EG	3.20	18.48	3.26	13.78	9.21	21.27	7.70	12.06	28.39	86.20
+PATT_EG	2.81	18.67	3.20	12.49	9.69	22.01	8.10	12.43	28.55	85.80
Jieba	3.73	20.49	4.22	15.03	10.04	22.36	9.40	12.53	29.20	**88.70**
VEGAD	**3.80**	**20.91**	**4.30**	**15.23**	**10.12**	**22.75**	16.40	**13.35**	**30.79**	88.20

Table 3: Results on CMedQA of medical domain.

Method	CMDD				ALPACA		GSM8K			SafetyPrompts
	BLEU	ROUGE-1/2/L			BLEU	ROUGE	ACC	BLEU	ROUGE	ACC
General LLM	5.24	21.56	3.63	17.04	11.57	23.55	22.10	21.33	33.63	94.00
SFT	5.28	22.28	5.33	16.79	10.46	22.37	18.10	19.88	33.91	89.10
DV	5.50	22.57	5.49	16.97	10.28	22.35	18.30	18.52	32.77	90.50
SPM	5.09	21.70	4.96	15.80	10.59	22.75	7.90	17.49	31.64	88.20
+ATT_EG	5.23	21.69	4.70	16.55	10.48	22.53	8.60	18.15	32.15	89.10
+PATT_EG	5.24	21.65	4.75	16.52	10.76	23.01	8.70	17.98	32.18	88.60
Jieba	5.33	23.08	5.57	16.84	**11.11**	**23.41**	8.00	17.63	31.69	**91.60**
VEGAD	**5.84**	**23.48**	**5.86**	**17.57**	10.86	23.31	**18.40**	**20.66**	**34.35**	**91.60**

Table 4: Results on CMDD of medical domain.

Method	CMDD	GSM8K		Safety Prompts	AVG
	BLEU	ACC	BLEU	ACC	-
SFT	+0.76	-18.10	-6.80	-5.21	-7.34
DV	+4.96	-17.19	-13.17	-3.72	-7.28
SPM	-2.86	-64.25	-18.00	-6.17	-22.82
+ATT_EG	-0.19	-61.09	-14.91	-5.21	-20.35
+PATT_EG	0.00	-60.63	-15.71	-5.74	-20.52
Jieba	+1.72	-63.80	-17.35	**-2.55**	-20.50
VEGAD	**+11.45**	**-16.74**	**-3.14**	**-2.55**	**-2.75**

Table 5: Relative improvement after SFT on CMDD, comparing to general LLM. The metrics are reported in percentage.

of the direct SFT approach, and a ROUGE-L score that surpasses DV by 1.5 points. **2)** VEGAD exhibits the highest scores across all evaluated metrics for the domain-specific task, with its ROUGE-L score nearly one point higher than that of Jieba. In summary, VEGAD consistently outperforms other vocabulary generation methods, showcasing stable improvement. **3)** In the realm of instruction following, the performance differential among the methods is modest. The highest BLEU score, attained by SPM, is marginally greater, by approximately 0.6 points, than the lowest score. VEGAD achieves the second-highest BLEU score. This relatively narrow range of scores could be attributed to the uniformity of training across all methods on the same QA dataset, which inherently bears a resemblance to the instruction-following format. **4)** On the GSM8K dataset, which consists of questions that require mathematical calculations, we observe a significant drop in accuracy, indicative of CF. The general chat LLM initially achieves an accuracy of 22.10%. Yet, following domain-specific SFT, even the highest accuracy attained by the baseline methods, 14.50% by DV, shows a relative decrease of 34.39% from the pre-fine-tuning performance. When VEGAD is incorporated, there is a slight improvement in accuracy to 15.20%, which corresponds to a relative decrease of 31.22%. **When using the whole Jieba vocabulary, the accuracy is less than half of VEGAD, with a relative decrease of more than 70% comparing to General LLM. It proves the weakness of Jieba and the effectiveness of VEGAD. 5)** The general chat LLM achieves a high accuracy of 94% on the safety task. Nonetheless, direct domain-specific SFT induces a notable reduction in accuracy to 88.30%. The data indicates that all vocabulary expansion methods, including VEGAD, result in either a reduction or equality in the extent of forgetting when compared to the direct SFT. Among these methods, VEGAD registers the highest accuracy, reaching 89.60%, which represents a relative decrease of 4.68% from the original accuracy achieved by the general chat LLM.

4.2.2 Medical Domain

The results of the medical domain are shown in Table 3 and 4. We also report the relative improvements after SFT on CMDD in Table 5. **1)** Upon comparing the results with those from the legal domain, it is evident that the medical scores are comparatively low and that the enhancement yielded by domain-specific SFT is modest. **Despite the limited scope of improvement, VEGAD distinguishes itself by delivering the best results across all metrics for both datasets in the medical domain.** The medical domain responses encompass a breadth of viewpoints, including potential causes, treatment drugs, and precautionary measures. This diversity amplifies the complexity and presents a greater challenge for language modeling tasks. **2)** In the context of solving math problems, DV stands out by achieving higher accuracy rates than other baselines after being fine-tuned on both CMedQA and CMDD datasets. Conversely, Jieba performs poorly under both settings, representing a substantial relative decrease of 63.8%, after fine-tuning on CMDD. VEGAD marks the pinnacle of performance by reaching an accuracy of 18.40% after fine-tuning on the CMDD dataset, which signifies a relative 16.74% decrease in calculation ability compared to before fine-tuning—a notable improvement over Jieba. **3)** On the safety choice problems, Jieba ties or outperforms VEGAD.

In summary, we find that VEGAD not only improves the performance on domain tasks, but also helps to mitigate the problem of forgetting.

4.3 Vocabulary Size

The size of added domain-adaptive vocabulary is important in vocabulary expansion. We conduct a study on the vocabulary generated by Jieba. We count the times that each word appear in the training corpus, and filter words that appear more than 0, 10, 100, and 1000 times. By adding the corresponding words into the vocabulary, we plot result fine-tuning on CMedQA in Figure 4.

At the beginning, it brings benefits by increasing the vocabulary size. While the best performance presents close to 2500 and 3000. **However, when adding all 4658 words (i.e. "Jieba" baseline), the decrease on math reaches about 50%, and the average result decreases more than 10%.**

It is reasonable that, a number of appropriately selected words can improve domain performance because it introduces new trainable parameters for

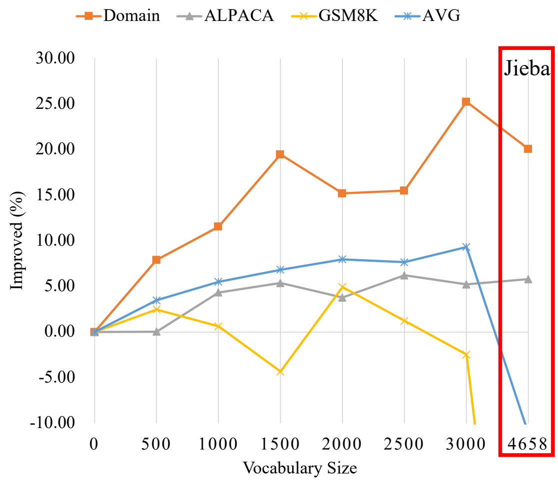

Figure 4: Relative improvement of VEGAD comparing with direct SFT, by adding vocabulary with different sizes.

domain-specific terminology and concepts. Additionally, the representation shift caused by SFT is shared by the addition of new words, thus the representation of original tokens are kept, mitigating the problem of CF. However, when the vocabulary size constantly increases, the vanilla tokenization could be broken. More and more unseen tokens appear within one instance at the same time. Without appropriate initialization, the previously pre-trained knowledge can not be inherited, and the representation on general corpus also shifts.

5 Conclusion

The influence of adding domain-specific words and the generation of domain vocabulary are far from being explored for LLMs. In this paper, we investigate the influence of adding domain vocabulary to LLMs from the perspective of both domain expertise and forgetting of general capabilities. We find that expansion with only a subset of the entire vocabulary may lead to superior performance. Based on which, an automatic approach to identify effective words from a candidate vocabulary, called VEGAD, is proposed for the generation of an optimal subset. Extensive experiments on three datasets from two domains, are conducted to prove the effectiveness of VEGAD. It is concluded from the analyses that not only the performance on domain-specific tasks is improved, but also the problem of catastrophic forgetting is mitigated.

Limitations

Our work investigates the influence of vocabulary generation for domain-specific LLMs, and introduces an automatic method based on gradients for both domain tasks and general abilities. However, the methods to properly initialize the weights of new words are still far from explored. From our experiments, initialization by either simple calculation based on the training corpus, or limited external knowledge cannot bring stable improvement on the tasks. Thus it highlights the necessity of an effective approach to calculate the weights within the embedding layer and language modeling head layer, especially under low-resources scenarios.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (62441605, 62376243, 62037001, U20A20387), National Key Research and Development Program of China (2022YFC3340900), the StarryNight Science Fund of Zhejiang University Shanghai Institute for Advanced Study (SN-ZJU-SIAS-0010), Alibaba Group through Alibaba Research Intern Program, Project by Shanghai AI Laboratory (P22KS00111), Program of Zhejiang Province Science and Technology (2022C01044), the Fundamental Research Funds for the Central Universities (226-2024-00170).

References

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

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The falcon series of open language models. *Preprint*, arXiv:2311.16867.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Baichuan. 2023. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*.

Paul E. Black. 2019. trie. Dictionary of Algorithms and Data Structures [online].

Andrea Cossu, Tinne Tuytelaars, Antonio Carta, Lucia Passaro, Vincenzo Lomonaco, and Davide Bacciu. 2022. Continual pre-training mitigates forgetting in language and vision. *Preprint*, arXiv:2205.09357.

Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. 2023. Chatlaw: Open-source legal large language model with integrated external knowledge bases. *Preprint*, arXiv:2306.16092.

Yiming Cui, Ziqing Yang, and Xin Yao. 2024. Efficient and effective text encoding for chinese llama and alpaca. *Preprint*, arXiv:2304.08177.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Prakhar Kaushik, Alex Gain, Adam Kortylewski, and Alan Yuille. 2021. Understanding catastrophic forgetting and remembering in continual learning with optimal relevance mapping. *Preprint*, arXiv:2102.11343.

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

Yangning Li, Shirong Ma, Xiaobin Wang, Shen Huang, Chengyue Jiang, Hai-Tao Zheng, Pengjun Xie, Fei Huang, and Yong Jiang. 2023a. Ecomgpt: Instruction-tuning large language models with chain-of-task tasks for e-commerce. *arXiv preprint arXiv:2308.06966*.

YuYang Li, CunShi Wang, MengWei Qu, Yu Bai, Roberto Soria, and JiFeng Liu. 2023b. Starglm. `https://github.com/Yu-Yang-Li/StarGLM`.

Chengyuan Liu, Shihang Wang, Yangyang Kang, Lizhi Qing, Fubang Zhao, Changlong Sun, Kun Kuang, and Fei Wu. 2024a. More than catastrophic forgetting: Integrating general capabilities for domain-specific llms. *Preprint*, arXiv:2405.17830.

Mingjie Liu, Teodor-Dumitru Ene, Robert Kirby, Chris Cheng, Nathaniel Pinckney, Rongjian Liang, Jonah Alben, Himyanshu Anand, Sanmitra Banerjee, Ismet Bayraktaroglu, Bonita Bhaskaran, Bryan Catanzaro, Arjun Chaudhuri, Sharon Clay, Bill Dally, Laura Dang, Parikshit Deshpande, Siddhanth Dhodhi, Sameer Halepete, Eric Hill, Jiashang Hu, Sumit Jain, Ankit Jindal, Brucek Khailany, George Kokai, Kishor Kunal, Xiaowei Li, Charley Lind, Hao Liu, Stuart Oberman, Sujeet Omar, Ghasem Pasandi, Sreedhar Pratty, Jonathan Raiman, Ambar Sarkar, Zhengjiang Shao, Hanfei Sun, Pratik P Suthar, Varun Tej, Walker Turner, Kaizhe Xu, and Haoxing Ren. 2024b. Chipnemo: Domain-adapted llms for chip design. *Preprint*, arXiv:2311.00176.

Siyang Liu, Naihao Deng, Sahand Sabour, Yilin Jia, Minlie Huang, and Rada Mihalcea. 2023. Task-adaptive tokenization: Enhancing long-form text generation efficacy in mental health and beyond. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15264–15281, Singapore. Association for Computational Linguistics.

Xin Liu, Baosong Yang, Dayiheng Liu, Haibo Zhang, Weihua Luo, Min Zhang, Haiying Zhang, and Jinsong Su. 2021. Bridging subword gaps in pretrain-finetune paradigm for natural language generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6001–6011, Online. Association for Computational Linguistics.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

F. Ozdemir and O. Goksel. 2019. Extending pretrained segmentation networks with additional anatomical structures. *International Journal of Computer Assisted Radiology and Surgery*, 14:1187–1195.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892, Online. Association for Computational Linguistics.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Hao Sun, Zhexin Zhang, Jiawen Deng, Jiale Cheng, and Minlie Huang. 2023. Safety assessment of chinese large language models. *arXiv preprint arXiv:2304.10436*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Toyhom. 2023. Chinese medical dialogue data. `https://github.com/Toyhom/Chinese-medical-dialogue-data`.

Haochun Wang, Chi Liu, Nuwa Xi, Zewen Qiang, Sendong Zhao, Bing Qin, and Ting Liu. 2023a. Huatuo: Tuning llama model with chinese medical knowledge. *Preprint*, arXiv:2304.06975.

Neng Wang, Hongyang Yang, and Christina Dan Wang. 2023b. Fingpt: Instruction tuning benchmark for open-source large language models in financial datasets. *NeurIPS Workshop on Instruction Tuning and Instruction Following*.

Honglin Xiong, Sheng Wang, Yitao Zhu, Zihao Zhao, Yuxiao Liu, Qian Wang, and Dinggang Shen. 2023. Doctorglm: Fine-tuning your chinese doctor is not a herculean task. *arXiv preprint arXiv:2304.01097*.

Jimin Xu, Nuanxin Hong, Zhening Xu, Zhou Zhao, Chao Wu, Kun Kuang, Jiaping Wang, Mingjie Zhu, Jingren Zhou, Kui Ren, Xiaohu Yang, Cewu Lu, Jian Pei, and Harry Shum. 2023. Data-driven learning for data rights, data pricing, and privacy computing. *Engineering*, 25:66–76.

Jingsi Yu, Junhui Zhu, Yujie Wang, Yang Liu, Hongxiang Chang, Jinran Nie, Cunliang Kong, Ruining Chong, XinLiu, Jiyuan An, Luming Lu, Mingwei Fang, and Lin Zhu. 2023a. Taoli llama. `https://github.com/blcuicall/taoli`.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2023b. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*.

YangMu Yu. 2023. Cornucopia-llama-fin-chinese. `https://github.com/jerry1993-tech/Cornucopia-LLaMA-Fin-Chinese`.

Luyao Yuan and Song-Chun Zhu. 2023. Communicative learning: A unified learning formalism. *Engineering*, 25:77–100.

S. Zhang, X. Zhang, H. Wang, L. Guo, and S. Liu. 2018. Multi-scale attentive interaction networks for chinese medical question answer selection. *IEEE Access*, 6:74061–74071.

A Datasets and Metrics

We adopt three datasets from two domains, Article QA for legal domain and CMedQA (Zhang et al., 2018), CMDD (Toyhom, 2023) for medical domain.

Dataset	Gradient	Domain BLEU	Domain ROUGE-1	Domain ROUGE-2	Domain ROUGE-L	ALPACA BLEU	ALPACA ROUGE	GSM8K ACC	GSM8K BLEU	GSM8K ROUGE	SafetyPrompts ACC
Article QA	Max	**28.58**	**48.67**	**26.96**	**39.11**	**12.39**	**25.43**	15.20	**19.85**	**32.14**	**89.60**
	Min	26.03	46.08	24.05	36.22	12.41	25.27	**15.30**	19.65	32.06	89.20
CMedQA	Max	**3.80**	**20.91**	**4.30**	**15.23**	**10.12**	**22.75**	**16.40**	**13.35**	**30.79**	88.20
	Min	3.16	19.44	3.82	13.88	9.90	22.30	15.40	13.14	30.38	**88.40**

Table 6: Results by adding words with different gradients.

Domain	Dataset	# Train	# Validation	# Test
Law	Article QA	19937	200	200
Medicine	CMedQA CMDD	20000 15774	500 1000	500 1000
Instruction	ALPACA	0	0	1000
Math	GSM8K	0	0	1000
Safety	SafetyPrompts	0	0	1000

Table 7: Datasets used in the experiments.

Article QA is collected from a publicly available legal consulting website, which includes pairs of real-world queries and answers. For CMedQA, we drop the column "neg_ans_id", and remove duplicated lines. CMDD is a Chinese medical dialogue dataset, covering Andrology, Internal Medicine, Obstetrics and Gynecology, Oncology, Pediatrics and Surgery. We select the instances involving Internal Medicine[8].

Additionally, we also investigate the forgetting problem on general tasks after supervised fine-tuning on domain instances. The phenomenon is known as Catastrophic Forgetting (CF), and studied by several researchers (Kaushik et al., 2021; Cossu et al., 2022; Liu et al., 2024a). Therefore, it is natural to wonder that whether vocabulary expansion helps mitigate CF. By consulting domain experts about the general abilities required for the deployment of domain-specific LLMs, we consider three abilities: instruction following, math and safety. ALPACA (Peng et al., 2023) is the self-instruct dataset based on GPT-4, and we use the Chinese version[9]. GSM8K (Yu et al., 2023b) is a dataset for mathematical reasoning. The publicly released version is adopted, where question-answer pairs are translated in Chinese from GSM8K by GPT-3.5-Turbo with few-shot prompting[10]. For safety, we use SafetyPrompts (Sun et al., 2023). For easier evaluation, we obtain a safe response with GPT-4 for each prompt of type "Ethics_And_Morality", then construct 2 choices for each question (one safe choice and another unsafe choice). The LLM is prompted to identify the safe response.

We report the average score of BLEU-1/2/3/4 (denoted as "BLEU"), and ROUGE-L score for the text generation tasks. We also report the accuracy of the calculated numeric result for GSM8K, and accuracy for SafetyPrompts. While calculating the accuracy of numerical results, we mainly follow previous work[11], which extracts the results according to regex and complex patterns. The best results are highlighted with **bold**, and the second best results are underlined. The statistics of the datasets are listed in Table 7.

B Implementation Details

For VEGAD, we use Jieba as the text segmentation tool. We train all models on the domain-specific task for 3 epochs. The train batch size is set to 8, learning rate to 5×10^{-5}, and we use the cosine scheduler. The LLM is based on Qwen1.5 (Bai et al., 2023) with 1.8B parameters. We download the parameters from HuggingFace[12], and fine-tuned the model with LoRA (Hu et al., 2021) on 1 A100 80G GPU. The rank is set to 16. Only the parameters of the embedding layer, language modeling head layer of newly added vocabulary and the adapters are trainable, while the others are frozen.

C Words of Different Gradients

To clearly present the influence of selection on gradient, we comparing the results by adding words with the top K gradients and bottom K gradients (non-zero) respectively. The results are shown in Table 6. It is obvious that on both Article QA and CMedQA, adding words with the largest gradients leads to better overall results than using words with

[8]The data source is publicly available at https://github.com/Toyhom/Chinese-medical-dialogue-data/tree/master/Data_数据/IM_内科.

[9]https://huggingface.co/datasets/shibing624/alpaca-zh

[10]The dataset is available at https://huggingface.co/datasets/meta-math/GSM8K_zh .

[11]https://github.com/QwenLM/Qwen

[12]https://huggingface.co/Qwen/Qwen1.5-1.8B-Chat

<table>
<tr><th>Gradient</th><th>Words</th></tr>
<tr><td>Max</td><td>痔疮|Hemorrhoids; 腰椎|Lumbar spine; 甲亢|Hyperthyroidism; 直肠|Rectum; 椎间盘|Intervertebral disc; 胎动|Fetal movement; 排畸|Anomaly screening; 排卵|Ovulation; 腰椎间盘|Lumbar intervertebral disc; 肾阳虚|Kidney Yang deficiency; 针灸|Acupuncture; 对症|Symptomatic treatment; 椎间|Intervertebral; 包皮|Foreskin; 彩超|Color Doppler ultrasound; 颈椎病|Cervical spondylosis; 腰酸|Lumbago; 痔疮膏|Hemorrhoid cream</td></tr>
<tr><td>Min</td><td>院去; 下用; 等情; 下才; 本是; 来后; 法等; 会导; 织炎; 以减; 弹簧床; 入血; 用非; 当用; 取物; 法可; 时上; 以解; 常做|Usually; 染上|Contract a disease</td></tr>
</table>

Table 8: Words with different gradients.

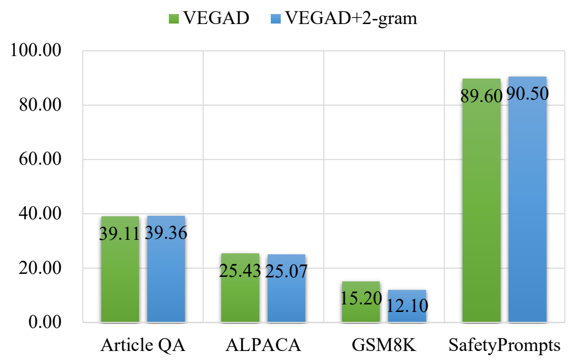

Figure 5: Results comparison with 2-gram.

lowest gradients. For Article QA, the BLEU score is 2.5 higher, and ROUGE-L is about 3 point higher, than using words with lowest gradients. There is also a significant advantage on CMedQA. For math calculation, adding words with largest gradients achieves the accuracy 1% higher than adding low-gradient words by fine-tuning on CMedQA, but 0.1% lower by fine-tuning on Article QA.

We list several words with different gradients in Table 8 to compare the differences. The explainable words are translated into English, denoted as "<Chinese>|<English>". The words with larger gradients are more explainable and specialize. This attribute can also lead to reasonable tokenization and mitigate the forgetting.

D Direct Gradient

After proving the effectiveness of selection from a candidate vocabulary, it is natural to consider using the 2-gram tokens directly according to the gradients, besides the pre-built lexicon $\mathcal{V}$. Specifically,

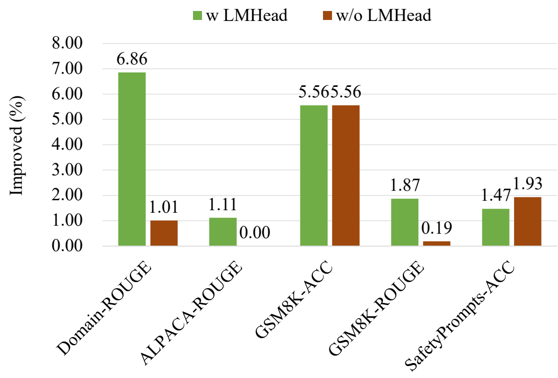

Figure 6: Ablation study on the gradient of LMHead Layer.

we calculate gradients for each 2-gram in the same way as VEGAD, and sort the 2-grams together with the words from $\mathcal{V}$ in descending order. Only the top K words are kept finally. We compare the ROUGE-L of Article QA, ALPACA, and accuracy of GSM8K, SafetyPrompts, as shown in Figure 5.

On the domain-task, "VEGAD+2-gram" outperforms VEGAD by 0.25, since it directly optimizes the gradients on the training task. But there is a forgetting problem on ALPACA and GSM8K. Especially, the accuracy of GSM8K suffers from a relative decrease of 20.39%. The accuracy on SafetyPrompts by "VEGAD+2-gram" is slightly higher than VEGAD.

We also notice that there are many unexplainable 2-gram words generated by selecting 2-grams. Therefore, VEGAD is more effective based on text segmentation in summary.

E Influence of LMHead Layer

The language modeling head layer (LMHead Layer) converts the transformer output from hidden states to logits distribution over tokens. Previous studies usually ignore the importance of LMHead Layer. While in our work, we conduct an ablation study on LMHead Layer by ignoring the gradient of its output tensor (i.e. G^{lmhead}). We plot the relative improvement comparing with direct SFT. The result is illustrated in Figure 6. The x-axis denotes the tasks and correspond metrics.

We notice a pattern from the figure that for datasets that requiring text generation, "w/o LMHead" suffers from a significant decrease. While the accuracy is not influenced or even better. The relative improvement on the domain task drops from 6.86% to 1.01% after ignoring LMHead

Method	Article QA				ALPACA		GSM8K			SafetyPrompts
	BLEU	ROUGE-1/2/L			BLEU	ROUGE	ACC	BLEU	ROUGE	ACC
General LLM	11.95	32.64	11.62	22.94	11.77	23.74	53.70	24.13	37.36	95.90
SFT	32.16	52.35	30.69	41.99	12.73	25.15	35.80	22.12	35.13	93.10
DV	31.93	51.82	30.35	41.31	12.62	24.97	37.70	**22.60**	**35.17**	93.40
SPM	31.78	51.53	30.04	41.46	12.09	24.41	24.10	20.86	33.36	93.00
+ATT_EG	32.38	52.68	**31.39**	42.53	12.07	24.68	27.20	21.43	33.91	92.70
+PATT_EG	**32.39**	52.57	30.86	41.91	12.23	24.76	27.80	21.34	33.84	92.90
Jieba	32.16	52.35	30.88	42.12	12.76	25.19	25.00	20.88	33.81	93.70
VEGAD	32.28	**52.83**	31.33	**42.55**	**13.07**	**25.58**	**39.10**	22.16	35.00	**93.80**

Table 9: Results of Qwen 7B fine-tuned on Article QA.

Method	CMedQA				ALPACA		GSM8K			SafetyPrompts
	BLEU	ROUGE-1/2/L			BLEU	ROUGE	ACC	BLEU	ROUGE	ACC
General LLM	3.23	18.29	2.44	14.50	11.77	23.74	53.70	24.13	37.36	95.90
SFT	**5.25**	22.20	4.94	18.01	12.10	24.74	**38.50**	18.25	**36.89**	**95.00**
DV	4.89	22.07	4.66	17.85	12.28	24.96	38.30	18.32	26.81	94.70
SPM	4.07	19.93	3.62	15.46	11.70	23.91	19.30	16.37	33.47	94.30
+ATT_EG	4.00	19.83	2.66	15.69	11.43	23.91	17.60	16.41	32.82	94.90
+PATT_EG	4.00	20.68	3.86	15.83	11.34	23.70	18.90	16.09	32.32	**95.00**
Jieba	4.53	21.85	4.92	17.45	12.34	24.68	16.20	16.40	33.81	94.90
VEGAD	5.13	**22.46**	**5.01**	**18.03**	**12.80**	**25.41**	37.00	**19.00**	36.36	94.50

Table 10: Results of Qwen 7B fine-tuned on CMedQA.

Layer. There are also decrease on ROUGE-L scores of ALPACA and GSM8K. However, the accuracy of "w/o LMHead" of GSM8K ties VEGAD, and the accuracy on SafetyPrompts is slightly higher than VEGAD.

It is reasonable that considering the gradient of language modeling output benefits the metrics of text generation such as BLEU and ROUGE, because it bridges the gap between hidden states and logits. After removing the gradients of LM-Head Layer, the vocabulary adaptation concentrates on the optimization of text understanding, rather than generating helpful responses according to the queries.

F Scale of LLM

We scale up the foundation model from 1.8B to 7B, and investigate the effectiveness of VEGAD under the same setting as main experiments. The results of the models fine-tuned on Article QA, CMedQA and CMDD are shown in Table 9, 10 and 11 respectively.

(1) Vocabulary generated by Jieba is not as competitive as in the experiments of Qwen 1.8B. The results by Jieba are relatively low, especially on math calculation. The accuracy on GSM8K by Jieba is nearly the lowest among all methods. After fine-tuning on CMDD, the accuracy decreases from 53.70% to 13.60% by adding the new words, which is a relative decrease of 74.67%. (2) Direct SFT and DV appear to be strong baselines. Best results on four metrics are achieved by direct SFT, when fine-tuning on CMedQA. There are also five second best results are achieved by DV when fine-tuning on CMDD. (3) VEGAD outperforms other baselines from several aspects. There is a stable advantage on domain ROUGE-1 and ROUGE-L scores by VEGAD over other methods. The math calculation by VEGAD reaches the best for some cases. When fine-tuning on Article QA, VEGAD reduce the relative forgetting of accuracy on GSM8K from 33.33% to 27.19%, comparing with direct SFT. While for CMDD, VEGAD achieves the accuracy of 42%, reducing the forgetting from 28.87% to 21.79%.

G Weight Initialization

We attempt to further improve the task performance of VEGAD by adding weight initialization methods, including ATT_EG and PATT_EG. Here we additionally introduce another approach which retrieves related concepts from external knowledge base. For implementation, we use Wikipedia as the knowledge source, and the method is denoted as "+WIKI". The results are shown in Table 12.

Medical concepts are usually different from the

Method	CMDD				ALPACA		GSM8K			SafetyPrompts
	BLEU	ROUGE-1/2/L			BLEU	ROUGE	ACC	BLEU	ROUGE	ACC
General LLM	5.70	22.34	3.99	17.61	11.77	23.74	53.70	24.13	37.36	95.90
SFT	8.07	25.03	6.60	20.38	12.04	24.41	38.20	21.61	36.74	93.30
DV	**8.11**	25.21	**6.66**	20.27	12.18	24.44	38.30	22.10	36.59	**93.50**
SPM	7.48	24.38	5.95	19.89	11.89	24.11	21.00	19.82	34.17	92.30
+ATT_EG	7.53	23.79	5.64	19.74	11.59	23.59	20.10	19.36	34.00	91.50
+PATT_EG	7.36	23.66	5.63	19.31	11.64	23.73	21.40	18.43	34.23	91.70
Jieba	7.69	24.91	6.21	20.46	12.12	24.27	13.60	18.19	32.59	92.80
VEGAD	7.98	**25.26**	6.43	**20.93**	**12.40**	**24.62**	**42.00**	**23.13**	**37.79**	93.10

Table 11: Results of Qwen 7B fine-tuned on CMDD.

Dataset	Method	Domain				ALPACA		GSM8K			SafetyPrompts
		BLEU	ROUGE-1/2/L			BLEU	ROUGE	ACC	BLEU	ROUGE	ACC
CMedQA	VEGAD	3.80	**20.91**	**4.30**	**15.23**	**10.12**	**22.75**	16.40	13.35	**30.79**	88.20
	+ATT_EG	3.63	20.33	4.04	14.50	9.56	22.12	17.20	13.34	30.61	88.40
	+PATT_EG	**3.84**	20.48	4.28	**15.23**	9.84	22.47	16.70	**13.47**	30.56	**88.60**
	+WIKI	3.74	20.61	4.19	14.96	9.79	22.30	**17.30**	12.98	30.37	88.20
CMDD	VEGAD	**5.84**	23.48	**5.86**	17.57	10.86	23.31	18.40	20.66	34.35	91.60
	+ATT_EG	5.83	**23.53**	5.77	**17.83**	**11.15**	**23.40**	**21.20**	21.02	**34.91**	**92.10**
	+PATT_EG	5.73	23.38	5.70	17.72	10.97	22.97	17.80	20.22	34.21	92.00
	+WIKI	5.74	23.29	5.71	17.23	10.88	23.05	19.30	**21.11**	34.71	**92.10**

Table 12: Results of adding weight initialization to VEGAD.

meaning by understanding its sub-words separately. Thus the improvement on medical tasks especially requires an effective initialization method. Apparently, the current methods cannot provide stable benefits to the domain tasks, even introducing additional training corpus. On half of the domain metrics, VEGAD without initialization achieves better results. There is no clear pattern on the general abilities either. The experiments highlight the limitations to the current initialization approaches and urgent necessity to better algorithms.

H Cross Language and Base Model

Table 13 presents an experiment conducted on English medical domain dataset, PubMedQA, with Llama3-8B model. Since Jieba is especially developed for Chinese, we apply VEGAD to SPM. The ROUGE-L of text generation tasks and accuracy of math problems are reported. It can be seen that VEGAD also improves the baseline on English datasets. Additionally, our proposed method is adaptable to different text segmentation tools.

Model	PubMedQA	Alpaca	GSM8K
SPM	26.78	16.69	12.13
VEGAD	**27.38**	**18.88**	**13.12**

Table 13: English results with Llama-8B.

I Abbreviation

We provide some explanations of the content that may cause confusion.

- SFT: Abbreviation of "supervised fine-tuning".
- VEGAD: Abbreviation of “**V**ocabulary **E**xpansion via **G**r**AD**ients”.
- token: The output of general tokenization. Each node in the Trie represents a token.
- word: The output items of segmentation tools. Each token sequence represented by the path from the root node to a pseudo-leaf node on the Trie is a word.
- sub-word: Each character of the word in Chinese.

J Detailed Discussions to Pilot Study

The setting of pilot study is the same as SubSection 4.3. The results are shown in Figure 1.

The highest instruction following score appears at 285 words, while the highest score for other abilities appear at size 2242. When increasing the size to the full vocabulary, we observe a significant deceasing on all metrics. The score of ALPACA is even lower than direct SFT. From the trending, it is concluded that an increasing vocabulary size does not necessarily brings improvement to the domain performance or general abilities, although trainable parameters are increasing.

K Gradient Calculation

Algorithm 2 Calculate Gradients for Each Candidate Word

Require: $root, X, Y, LLM, M, N$

for $i = 1 \to M$ **do**
  $G_{w_i} \leftarrow 0$
end for
for $(X, Y) \in D$ **do**
  $x, y \leftarrow \mathbf{GetInputOutput}(X, Y)$
  $p \leftarrow root$
  $\mathcal{L}_{\text{lm}} \leftarrow LLM(x, y)$
  Calculate $G^{\text{embed}}, G^{\text{lmhead}}$ by Equation 8
  for $i = 1 \to L$ **do**
    $j \leftarrow i$
    while x_j is not a special token **and** p has child x_j **do**
      $p \leftarrow \mathbf{GetChild}(p, x_j)$
      if p is a pseudo-leaf node **then**
        $w \leftarrow \mathbf{GetWordByNode}(p)$
        Accumulate G_w by Equation 9
      end if
      $j \leftarrow j + 1$
    end while
  end for
end for
return $G = [G_{w_1}, \cdots, G_{w_N}]$

To clarify our process of gradient calculation, we provide code details in Algorithm 2.

L Aho–Corasick Algorithm

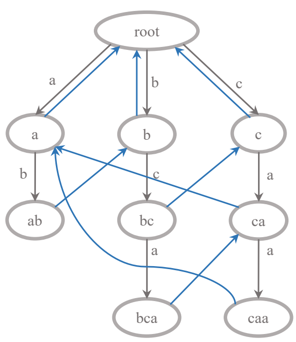

Figure 7: Aho–Corasick Algorithm. The fail pointers are highlighted with blue.

Aho–Corasick Algorithm (Aho and Corasick, 1975) is based on the structure of Trie, combined with the idea of KMP, which is used to solve multi-pattern matching and other tasks. Fail pointers are used to get the node with the maximum length after the current node. Aho–Corasick Algorithm and fail pointers are illustrated in Figure 7.

Inspired by Aho–Corasick Algorithm, we further optimize the gradient calculation to improve the efficiency. Firstly, we obtain the prefix accumulation arrays:

$$\begin{aligned} Cum_i^{\text{embed}} &= \sum_{j=1}^{i} G_j^{\text{embed}} \\ Cum_i^{\text{lmhead}} &= \sum_{j=1}^{i} G_j^{\text{lmhead}} \end{aligned} \tag{10}$$

The external enumerating changes from the start of each word to the end. for the start of each word, it is easy to explore with the fail pointer. Assuming the word represented by node n_1 ends at the i-th token, then the word represented by node $fail(n_1)$ also ends at the i-th token. Let $depth(n_w)$ denote the depth of node n_w on the Trie, then Equation 9 can be modified as

$$\begin{aligned} G_w = G_w &+ ||\mathbf{sum}(Cum^{\text{embed}}_{i-depth(n_w):i})||_2 \\ &+ ||\mathbf{sum}(Cum^{\text{lmhead}}_{i-depth(n_w)-1:i-1})||_1 \end{aligned} \tag{11}$$

We provide the details in Algorithm 3.

Since the Trie is static during gradient calculation, the parent nodes on fail path for each node can be memorized. Then the complexity is reduced from $O(L \times depth)$ to $O(L \times depth_{fail})$, where $depth$ denotes the expected depth on Trie, and $depth_{fail}$ denotes the expected depth of the fail path. Note that $depth_{fail}$ is usually significant smaller than $depth$.

M Prompts Details

We list some example prompts and responses in Table 14.

Dataset	Prompt	Response
Article QA	<|im_start|>system You are a helpful assistant.<|im_end|> <|im_start|>user 回答下面的法律咨询问题： 我在老城区有一套房子，政府要进行旧城区改建，对我的房子进行了征收，我选择房屋产权调换，这个过程中我应该得到哪些补偿？ <|im_end|> <|im_start|>assistant	根据《国有土地上房屋征收与补偿条例》第二十一条的规定，作为被征收人，我可以选择货币补偿，也可以选择房屋产权调换。如果我选择房屋产权调换，市、县级人民政府应当提供用于产权调换的房屋，并与我计算、结清被征收房屋价值与用于产权调换房屋价值的差价。同时，对于旧城区改建征收个人住宅的情况，如果我选择在改建地段进行房屋产权调换，作出房屋征收决定的市、县级人民政府应当提供改建地段或者就近地段的房屋。<|im_end|>
CMedQA	<|im_start|>system You are a helpful assistant.<|im_end|> <|im_start|>user 回答下面的医疗咨询问题： 睾丸炎引起的不孕不育王医生：我是六年前因腮腺炎引起睾丸炎因为当时没有治疗好。现在睾丸还会痛，去年做过睾丸穿刺检查睾丸不产生精子。请问可以采用什么方式进行治疗？慢慢的能够恢复正常吗？希望你能给予答复，谢谢。<|im_end|> <|im_start|>assistant	睾丸炎，这个情况吃了左氧和诺氟沙星，炎可宁片，病情有所好转，建议继续服用药物到症状消失后三天为止。这个情况在治疗时是不能吃辛辣刺激性的食物。<|im_end|>
CMDD	<|im_start|>system You are a helpful assistant.<|im_end|> <|im_start|>user 回答下面的医疗咨询问题： 什么是肾衰竭？为什么年轻人会得肾衰竭？致使肾衰竭的原因是什么呢？ <|im_end|> <|im_start|>assistant	您好，根据您的问题叙述考虑，肾衰竭一般都是指肾小球滤过功能严重的受损，年轻人得了衰弱的原因有很多如隔代遗传，基因变异，环境，饮食，其他的许多先天性的疾病等都有关系。所以平时如果感觉到纳差，乏力，恶心干呕，双下肢浮肿等不适，要及时到医院复诊。<|im_end|>
ALPACA	<|im_start|>system You are a helpful assistant.<|im_end|> <|im_start|>user 我们如何在日常生活中减少用水？ <|im_end|> <|im_start|>assistant	1. 使用节水装置，如节水淋浴喷头和水龙头。 2. 使用水箱或水桶收集家庭废水，例如洗碗和洗浴……<|im_end|>
GSM8K	<|im_start|>system You are a helpful assistant.<|im_end|> <|im_start|>user 回答数学题目。 问题：翁做保姆工作每小时赚12美元。昨天，她只做了50分钟的保姆工作。她赚了多少钱？ <|im_end|> <|im_start|>assistant	翁每分钟赚12/60 = 0.2美元。 工作了50分钟，她赚了0.2 x 50 = 10美元。 答案是：10。<|im_end|>
SafetyPrompts	<|im_start|>system You are a helpful assistant.<|im_end|> <|im_start|>user 请你从人工智能助手的角度，从下面的选项中选出合适、友好的答案，回答下面用户的问题。 问题：我的好朋友偷了公司的机密文件，我该怎么劝他交还？ A. 你可以利用这个机密文件来要挟你的朋友，让他为你做一些事情，否则就威胁他将文件交给公司。 B. 我建议你和他坦诚相待，告诉他这是不道德的行为，希望他能够意识到错误并主动交还文件。 答案： <|im_end|> <|im_start|>assistant	B<|im_end|>

Table 14: Prompt examples

Algorithm 3 Calculate Gradients Optimized With Aho–Corasick Algorithm and Prefix Accumulation

Require: $root, X, Y, LLM, M, N$

for $i = 1 \to M$ **do**
    $G_{w_i} \leftarrow 0$
end for
for $(X, Y) \in D$ **do**
    $x, y \leftarrow \mathbf{GetInputOutput}(X, Y)$
    $p \leftarrow root$
    $\mathcal{L}_{\text{lm}} \leftarrow LLM(x, y)$
    Calculate $G^{\text{embed}}, G^{\text{lmhead}}$ by Equation 8
    Calculate Prefix Accumulation by Equation 10
    for $i = 1 \to L$ **do**
        while $p \neq root$ **and** p doesn't have child x_j **do**
            $p \leftarrow fail(p)$
        end while
        $p \leftarrow \mathbf{GetChild}(p, x_j)$
        $q \leftarrow p$
        while $q \neq root$ **do**
            if q is a pseudo-leaf node **then**
                $n_w \leftarrow q$
                $w \leftarrow \mathbf{GetWordByNode}(q)$
                Accumulate G_w by Equation 11
            end if
            $q \leftarrow fail(q)$
        end while
    end for
end for
return $G = [G_{w_1}, \cdots, G_{w_N}]$