Unifying Demonstration Selection and Compression for In-Context Learning

Anonymous ACL submission

Abstract

In-context learning (ICL) enables LLMs to exhibit spectacular emergent capabilities in various scenarios. Unfortunately, introducing demonstrations easily makes the prompt length explode, bringing a significant burden to hardware. In addition, random demonstrations usually achieve limited improvements in ICL, necessitating demonstration selection among accessible candidates. Previous studies introduce extra modules to perform demonstration compression or selection independently. In this paper, we propose an ICL framework UniICL, which **Uni**fies demonstration selection and compression, and final response generation via a single frozen LLM. UniICL leverages the understanding ability of well-trained LLMs to independently compress different demonstrations into compressed features, and then a learnable projection layer converts features to LLM-acceptable compressed virtual tokens. Apart from substituting original demonstrations to reduce input length, virtual tokens are again used to select potential demonstrations. Finally, current queries together with selected compressed virtual tokens are fed into the same frozen LLM for response generation. UniICL is a parameter-efficient framework that only contains 17M trainable parameters originating from the projection layer and a learnable embedding. We build UniICL upon two backbones and conduct experiments over in- and out-domain datasets of both generative and understanding tasks, encompassing ICL scenarios with plentiful and limited demonstration candidates. Results show that UniICL effectively unifies 12× compression, demonstration selection, and response generation, efficiently scaling up the baseline from 4-shot to 64-shot ICL with 24 GB CUDA allocation[1].

[1]The code and model will be released in the final version.

1 Introduction

In-context learning (ICL) (Brown et al., 2020; Xie et al., 2021; Wang et al., 2023b) exhibits powerful performance in practical applications with the emergence of scaled-up Transformer-based (Vaswani et al., 2017) Large Language Models (LLMs) (Wang et al., 2023c; Yang et al., 2023; Wei et al., 2023; Wang et al., 2023a; Min et al., 2022). In ICL, the provided demonstrations activate the pre-training knowledge of LLMs to allow them to perform well on various downstream tasks such as text summarization (Wang et al., 2023c; Yang et al., 2023) and text classification (Min et al., 2022) without gradient updating of billion parameters. Despite its significant role in the era of LLMs, ICL also brings an enormous challenge to the input window. Specifically, inevitably introducing demonstrations directly causes length explosion (Wang et al., 2024), bringing significant memory costs and decreasing inference throughput as described in Figure 8. Except for length explosion, Liu et al. (2021) points out that the quality of the selected demonstrations significantly influences the ICL performance.

Recent efforts in modifying model architecture significantly expand the input window (Zheng et al., 2022; Wu et al., 2022; Ding et al., 2023; Bulatov et al., 2023). However, LLMs with million context windows still struggle to overcome performance degradation (Liu et al., 2024). Researchers also attempted to alleviate length explosion via prompt pruning (Jiang et al., 2023) or soft prompts (Wingate et al., 2022; Mu et al., 2023; Ge et al., 2023). Their main idea is to train an independent compressor to compress the input into soft prompts. While in selecting powerful in-context demonstrations, Liu et al. (2021); Lu et al. (2021) scores each candidate demonstration for the current queries via an extra ranker, and Ram et al. (2023); Wang et al. (2024) finds that a fine-tuned LLM is up to the task

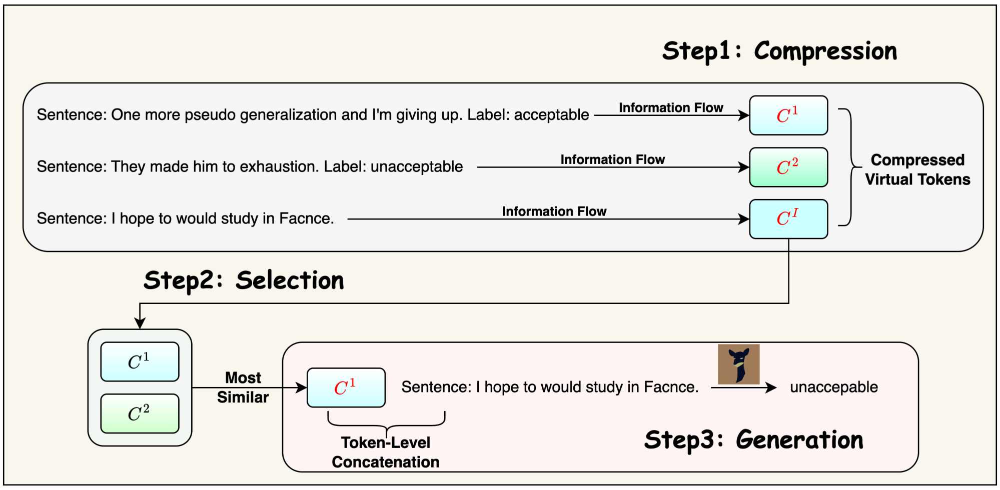

Figure 1: UniICL compresses candidate demonstrations into spans of compressed virtual tokens and selects demonstrations similar to the query. Virtual tokens substitute the original demonstration together with the query, fed to the same LLM for prediction generation.

of ranking.

However, the standalone compressor or ranker inevitably incurs additional memory costs as it necessitates simultaneous loading alongside the target LLM. To tackle these challenges, we propose a framework UniICL as illustrated in Figure 1, which leverages the semantic understanding ability of target LLMs developed during pre-training to compress and select demonstrations. UniICL keeps the target LLM frozen during training to avoid catastrophic forgetting and reduce training costs. Considering the frozen LLM within UniICL function as both compressor and generator, UniICL is training efficiently and does not need to load extra compression modules, significantly reducing memory costs for both training and inference. As the underlying LLMs in visual language models fail to understand visual features without an adapter, UniICL still requires converting compressed features into LLM-acceptable compressed virtual tokens. Apart from substituting lengthy demonstrations, virtual tokens are dense representations of original demonstrations, which can be naturally further applied to select potential demonstrations by measuring similarity. In this paper, the connector is a projection layer supervised-tuned under the language modeling objective of the generator during learning compression and jointly optimized by language modeling and contrastive objectives (He et al., 2020) during learning selection.

UniICL notices the fact that demonstrations in the ICL are independent of each other, which is ignored by previous compression studies (Mu et al., 2023; Wingate et al., 2022; Ge et al., 2023). Therefore, UniICL independently compresses demonstrations and proposes to configure Demonstration Bank (DB) to cache compressed virtual tokens for further reusing without requiring repeated compression for the same demonstration. We evaluate UniICL which only contains 17M trainable parameters on a scope of benchmarks involving linguistic acceptability, semantic classification, text summarization, and passage reranking, and UniICL achieves outstanding performances. Then, the in-domain experiments show that UniICL built upon two backbones (Vicuna and BlueLM) effectively substitutes the 12× demonstrations with soft prompts, easily scaling up the inner LLM from 4-shot to 64-shot. Our main contributions are as follows:

- To our knowledge, we are the first to propose an ICL framework that unifies compression, selection, and generation via a single frozen LLM.
- UniICL is a memory-friend framework that enables LLMs to perform large-shot ICL on consuming GPUs.
- UniICL proposes to configure Demonstration Bank to enhance ICL efficiency, avoiding repeated compression for the same demonstration.

2 Related Work

2.1 Soft Prompt Compression

Recently, researchers attempted to utilize soft prompts to convert actual tokens to dense-information virtual tokens. Mostly from a distillation perspective, Wingate et al. (2022) aligned the teacher model and the student model, where the teacher model accepted the actual task instruction while the student model fed the soft prompt. The main drawback of this approach was the lack of generalization that necessitated training for each lexically different instruction. To tackle the generalization problem, Mu et al. (2023) proposed to learn a Llama-7b to compress instruction to virtual tokens, but only compress instruction was not powerful enough since the demonstrations were much longer in practice. To compress the demonstrations, Chevalier et al. (2023) proposed AutoCompressor to recurrently generate compressed virtual tokens based on a fine-tuned Llama (Zhang et al., 2022). However, AutoCompressor broke the independence of demonstrations, and the recurrent compression increased inference latency. Ge et al. (2023) proposed ICAE that employed a LoRA-adopted Llama-7b (Touvron et al., 2023) to compress the processed demonstrations to compact virtual tokens, while ICAE still struggled to overcome quite long inputs.

2.2 Extractive Compression

Apart from employing soft prompts, researchers also endeavored to shorten prompts by extracting informative tokens from the original ones (Li, 2023; Jiang et al., 2023), namely token pruning (Kim et al., 2022) or token merging (Bolya et al., 2022). Recent works like LLMLingua (Jiang et al., 2023) and Selective Context (Li, 2023) shared similarities but diverged on whether to eliminate tokens with high or low Perplexity (PPL). LLMLingua emphasized tokens with high PPL, attributing them as more influential, resulting in achieving outstanding performance. As mentioned in their paper, extractive compression methods encountered Out-of-Distribution (OoD) issues between the extractor and the target LLM. To reconcile this, they fine-tuned Alpaca-7b (Taori et al., 2023) using the Alpaca dataset (Taori et al., 2023) to perform the alignment. However, UniICL naturally bypassed the distribution-aligned module, since the compressor and the target LLM were the same.

Methods	Additional Compressor	Compression Tool	# Trainable Parameters	Train Size
LLMLingua	YES	Pruning	7B	57k
AutoCompressor	NO	Soft Prompt	7B	UNKNOWN
ICAE	YES	Soft Prompt	70M	240k
UniICL	NO	Soft Prompt	17M	30k

Table 1: Comparsion among recent compression methods and UniICL. Compression Tool represents the involved compression technique of different methods.

3 Methodology

We propose UniICL, a parameter-efficient ICL framework that unifies demonstration compression, demonstration selection, and response generation via a single LLM. As for the selection of the underlying LLM, previous work has proved that the Decoder-only model performs better than the Encoder-Decoder model in prompt compression (Mu et al., 2023). We follow this conclusion and employ popular Vicuna-7b (Zheng et al., 2023) and BlueLM-7B[2] (Team, 2023) as the underlying backbone in UniICL.

We present a comparison of training costs between UniICL and other recent compression methods in Table 1. Additionally, we illustrate differences in formulating virtual tokens for compression methods based on the soft prompt in Figure 2.

To explain plainly, we ideally assume the compressor within three compression methods based on soft prompts has the window limitation of L, and has the same compression ratio, ignoring the length of soft prompts. In the 2-shot scenario, demonstrations D_1 and D_2 have a length of L. Considering the AutoCompressor, the concatenated demonstrations will be divided back into two segments, and the AutoCompressor compresses each segment step-by-step, bringing two times non-parallel compression. When it comes to ICAE, merely D_1 is accessible for the compressor and others will be read by no means. AutoCompressor shows advantages in the readable prompt length, but is short in efficiency, while ICAE has a constant compression complexity but struggles to approach relatively long inputs. Additionally, AutoCompressor makes the compression of later demonstrations be conditioned on the previous ones, which breaks demonstration independence in ICL.

Combining the advantages of AutoCompressor and ICAE, UniICL compresses demonstrations independently and introduces virtual tokens concatenation to overcome long prompt challenges. In the N-shot settings, the number of practical compres-

[2]We mainly discuss UniICL built on Vicuna-7B, the experiments of BlueLM will be exhibited in Appendiex B.

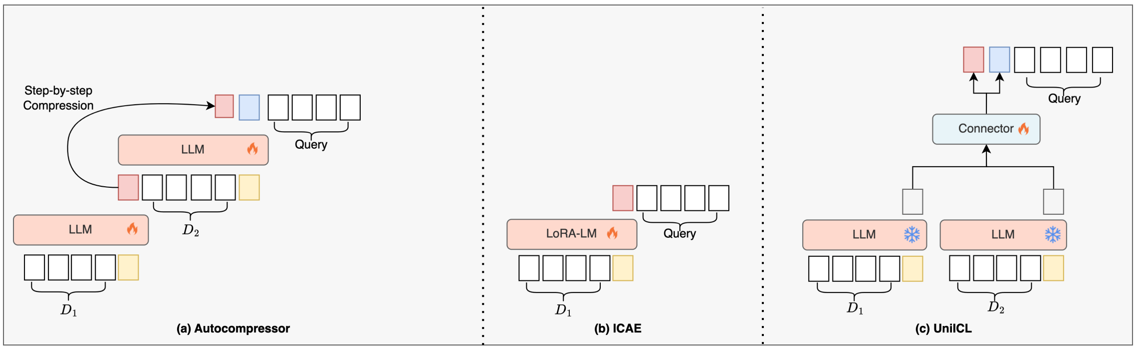

Figure 2: Differences of compression methods in formulating compressed virtual tokens in ICL.

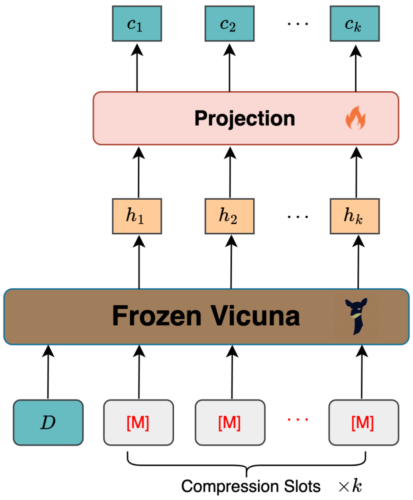

Figure 3: Demonstration compression.

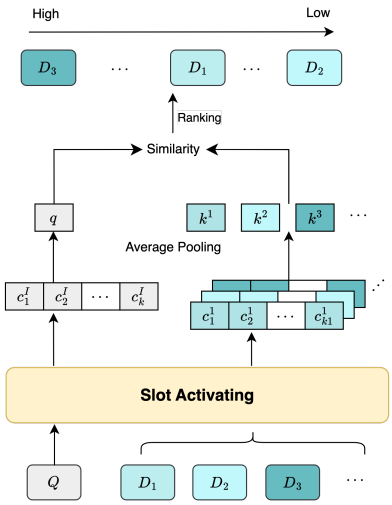

Figure 4: Demonstrations selection.

sion steps can be calculated as $\lceil \frac{N}{k} \rceil$, where k indicates that a single GPU is capable of compressing k demonstrations in a batch. When the GPU capacity is sufficient, k equals N, which is the scenario of ICAE that compresses all segments in a time but UniICL drops nothing, while it degenerates to the AutoCompressor scenario that compresses segments step-by-step, when the GPU capacity is only sufficient to set $k = 1$.

3.1 Demonstration Compression

UniICL introduces memory slots [M] $\in \mathcal{R}^d$, a learnable d-dimension embedding initialized from a rarely used embedding of the target LLM. UniICL activates the compression slots to absorb information from demonstrations in the forward propagation of frozen Vicuna, as illustrated in Figure 3. We first attach k compression slots $M = k \times$ [M] to each demonstration D_i, formatting modified prompt fed to the Vicuna. Then, frozen Vicuna forwards the modified prompts and outputs the last hidden states $H_i = (h_1, h_2, ..., h_k)$ on top of the k compression slots, dropping the others[3]:

$$\_, H^i = \text{forward}(D_i \oplus M) \tag{1}$$

Due to the attention mechanism, H^i is compelled to attend to the preceding actual tokens. Then, UniICL inserts a projection layer to convert H^i into LLM-acceptable compressed virtual tokens $C^i = (c^i_1, c^i_2, ..., c^i_k)$:

$$c^i_j = W_p \cdot h^i_j, \tag{2}$$

where W_p is the parameters of the projection layer.

3.2 Demonstration Selection

Except for substituting the origin demonstrations for generation, compressed virtual tokens C^i are representative of demonstrations that can be again applied for demonstration selection, as illustrated

[3] $\oplus$ means token-level concatenation.

in Figure 4. Specifically, given a query Q and its candidate demonstrations $(D_1, D_2, ..., D_n)$, UniICL obtains their latent representation after average pooling:

$$\bar{C^i}_{Q/D} = \frac{1}{k}\sum_{j=1}^{k} c_j. \tag{3}$$

We define the i-th demonstration saliency score S_i via the cosine similarity between Query and demonstrations:

$$S_i = \cos(\bar{C_Q}, \bar{C_D^i}). \tag{4}$$

Demonstrations are then reranked according to their saliency scores.

3.3 In-context Generation

We employ the frozen Vicuna again to generate responses with the guiding of concatenated virtual demonstrations and queries, as illustrated in Figure 5. For m-shot in-context learning, we obtain m spans of virtual tokens after demonstration compression and selection, denoted as C^1 to C^m. Then, we horizontally concatenate them, keeping their relative position unmodified. Finally, the concatenated virtual tokens together with actual inference inputs are fed into Vicuna, performing auto-regressive generation as normal:

$$y_i = \text{generate}(C^1, ..., C^m; Q; y_{<i}) \tag{5}$$

Except for the generative manner, virtual tokens are conveniently transferred to close-ended evaluation for understanding tasks through providing the candidate answers and measuring the label space PPL [4], e.g. (ppl^+, ppl^-) for sentiment classification:

$$y = \text{argmin}(ppl^+, ppl^-), \tag{6}$$

where answers with PPL closest to 1 are judged to be the current prediction.

3.4 Training

The trainable parameters in UniICL are merely 17M originating from the projection layer W_p and the introduced compression slot [M]. The projection layer is optimized with the language modeling objective of Vicuna to learn a base compression model. Then InfoNCE (He et al., 2020) joint with language modeling objective are used to augment the demonstration selection ability of the base compression model:

$$\mathcal{L} = \mathcal{L}_{lm} + \mathcal{L}_{ctr}. \tag{7}$$

Specifically, we slice the source input of each training instance into two parts and randomly compress one, denoted the compression part as x_c and the unmodified as x_u. Afterward, we attach M to x_c and get virtual tokens C on top of the compression slots, as described in Equ. 1 and Equ. 2. Therefore, the language modeling loss $\mathcal{L}_{lm}$ is obtained as:

$$\mathcal{L}_{lm} = -\frac{1}{|y|}\sum_{t=0} logP(y_t|Q; C; y_{<t}), \tag{8}$$

where y is the reference label of the current training instance. Additionally, to approach the large-shot settings without significant truncation, we introduce concatenation compression. When x_c exceeds the window limitation for compression, UniICL further divides x_c into acceptable ranges and compresses them independently to get local virtual tokens. Then, these virtual tokens of several segments will be concatenated to formulate global virtual tokens to replace x_c.

Consequently, we utilize contrastive learning for selection augmentation and mine positives and negatives as illustrated in Figure 6. Specifically, given each training instance Q and n candidate demonstrations $(D_1, D_2, ..., D_n)$ from two non-crossing training subsets, we employ Vicuna to calculate the PPL concerning the golden label of Q, denoted as ppl^Q. Then, we provide the i-th demonstration and calculate PPL concerning the golden label of Q, denoted as $(ppl_i^D, i \in [1, n])$. We count ppl^Q as the baseline and calculate candidate relative PPL gains:

$$\widetilde{ppl_i^D} = ppl^Q - ppl_i^D, i \in [1, n]. \tag{9}$$

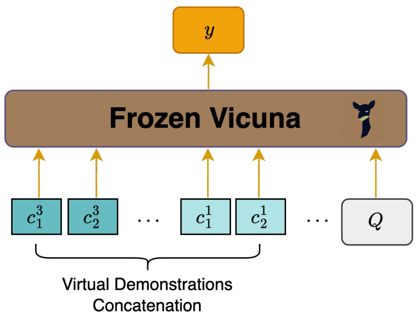

Figure 5: In-context generation.

[4] https://huggingface.co/docs/transformers/perplexity

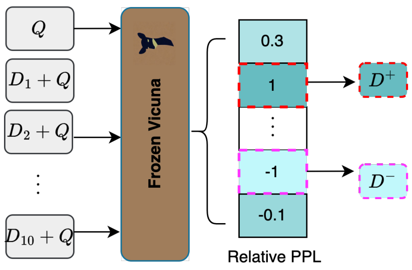

Figure 6: Negatives mining pipeline.

Hence, C_D^+ (C_D^-) is the representation of demonstrations D^+ (D^-) that furthest reduces (increases) ppl^Q as processed in Equ. 3, and the contrastive loss $\mathcal{L}_{ctr}$ can be formulated as:

$$\mathcal{L}_{ctr} = \frac{exp(cos(C_Q, C_D^+))}{exp(cos(C_Q, C_D^+)) + exp(cos(C_Q, C_D^-))}. \tag{10}$$

In particular, if all relative PPL gains are less than 0, namely none of the candidate demonstrations help guide Vicuna to generate the golden label, we will apply the other set of candidates.

4 Experiment

4.1 Baselines

Naive Vicuna-7b serves as the fundamental baseline fed into actual demonstrations. AutoCompressor recurrently compresses demonstrations into virtual tokens. We employ their Llama2-7b version[5]. LLMLingua is a coarse-to-fine demonstration pruning method based on dropping uninformative words. We employ their released 7b version[6], of which compressor is a fine-tuned Llama-2. For a meaningful comparison, **we replace target LLMs of LLMLingua (GPT-3.5-Turbo or Claude-v1.3) with the Vicuna-7b.** ICAE[7] compresses demonstrations into soft prompts via a LoRA-adapted Llama2-7b.

Additionally, since selection augmentation is involved in the training of UniICL, we utilize the popular Sentence-BERT (S-BERT) (Reimers and Gurevych, 2019) as the dense retriever to construct an ICL pipeline for the above methods, serving as simple but effective selection-based baselines.

[5]`https://github.com/princeton-nlp/AutoCompressors`.

[6]`https://github.com/microsoft/LLMLingua`.

[7]`https://github.com/getao/icae`.

4.2 Settings

Considering the involved datasets and computation efficiency, we set the max allowed input length limit to 512 for both compression and generation. For a fair comparison, we set the allowed window of baselines to 512 and their compression ratio to the same as UniICL. We fix the learning rate to 8e-5 and use Adam as the optimizer, and the effective batch size is 32 (8 GPUs data parallelism and 4 steps gradient accumulation). Additionally, we conducted all experiments on 8*NVIDIA A5000 24G GPUs based on BFloat 16 data type, and we set the evaluated shot to 8 for understanding tasks and 5 for generative tasks for illustration because of marginal ICL gains and memory costs.

We apply S-BERT to pre-rank and output the top 10 similar candidates from training sets according to each inference input for all baselines. UniICL is employed to perform selection among them in practice due to computation efficiency for high-resource ICL. On the contrary, the low-resource ICL setting fixes the random candidate demonstrations to 20 for all inference inputs, performing pre-ranking and selecting as well.

4.3 Results

We comprehensively evaluate the ICL performance of UniICL on the out-domain dataset CoLA, SST-2, and IMDb by close-ended evaluation and ARXIV by open-ended evaluation, as demonstrated in Table 2. Specifically, UniICL outperforms naive Vicuna-7b fed with actual candidate demonstrations, which indicates that virtual demonstrations are more efficient and informative for guiding the target LLM, and UniICL outperforms all the baselines by compressing the same demonstrations preranked by S-BERT. Additionally, UniICL achieves further performance gains after selecting demonstrations via itself. Additionally, open-ended results indicate that virtual demonstrations still efficiently capture semantic information for ICL guiding, even though summarization demonstrations are much longer than understanding ones. Regarding ARXIV, the original ICL is not helpful enough due to its extremely over-length document, leaving little room for demonstrations. UniICL works as expected by compressing demonstrations and concatenating virtual demonstrations, and achieves +2.8 R-1 gains in the 5-shot setting that selects demonstrations via selection-augmented virtual tokens.

Model	#-shots	CoLA-dev	SST-2-dev Acc.	IMDb	ARXIV R-1	ARXIV R-2	ARXIV R-L	XSum R-1	XSum R-2	XSum R-L
Vicuna	0-shot	56.2	91.7	92.6	34.3	9.1	27.4	19.9	5.0	13.5
	1-shot	58.2 (57.4)	90.7 (90.8)	91.9 (91.0)	34.4 (33.2)	9.1 (8.5)	27.5 (26.7)	21.2 (20.4)	5.8 (5.2)	14.5 (13.9)
	2-shot	62.1 (59.8)	92.1 (91.3)	91.7 (91.7)	-	-	-	-	-	-
	5-shot	62.3 (61.9)	93.0 (91.9)	94.1 (92.5)	-	-	-	-	-	-
AutoCompressor	1-shot	42.1 (40.9)	85.7 (84.2)	95.0 (95.1)	27.0 (26.4)	8.4 (8.2)	26.1 (25.8)	21.3 (20.3)	6.5 (6.3)	13.7 (13.7)
	2-shot	58.8 (56.3)	88.0 (86.4)	95.0 (94.6)	27.1 (26.2)	8.6 (7.9)	26.4 (25.4)	21.9 (21.4)	6.6 (6.4)	14.5 (14.1)
	5-shot	59.1 (58.8)	91.3 (89.1)	94.7 (94.8)	34.5 (33.7)	9.4 (9.1)	28.7 (27.9)	22.4 (21.7)	6.9 (6.7)	14.8 (14.3)
LLMLingua	1-shot	55.5 (55.0)	89.7 (89.6)	91.0 (89.9)	33.3 (33.1)	8.9 (8.7)	27.4 (27.1)	20.5 (19.7)	5.4 (5.2)	14.5 (14.4)
	2-shot	56.7 (55.7)	90.7 (90.2)	91.3 (91.0)	32.9 (32.0)	8.2 (8.1)	26.9 (25.9)	20.3 (20.0)	5.2 (5.1)	14.3 (14.1)
	5-shot	57.2 (56.9)	90.6 (90.2)	90.9 (91.2)	30.1 (29.7)	7.9 (7.4)	25.3 (24.6)	19.7 (18.6)	4.9 (4.9)	14.1 (14.3)
ICAE	1-shot	30.9 (30.9)	61.0 (60.1)	85.7 (83.3)	26.8 (24.6)	8.2 (7.1)	24.7 (22.9)	23.5 (21.9)	8.5 (7.8)	20.9 (20.3)
	2-shot	30.9 (30.9)	49.0 (52.8)	85.9 (85.9)	27.2 (25.5)	8.4 (7.6)	25.9 (24.3)	24.4 (23.2)	8.9 (8.4)	21.3 (20.8)
	5-shot	30.9 (30.9)	54.2 (51.0)	85.7 (85.9)	28.3 (26.9)	8.7 (7.7)	26.6 (25.8)	25.3 (24.9)	9.2 (8.8)	22.5 (21.6)
UniICL	1-shot	58.7 (58.0)	92.9 (91.7)	94.3 (92.3)	35.5 (34.7)	10.5 (10.2)	28.7 (27.9)	27.7 (25.5)	10.2 (9.1)	21.2 (20.0)
	2-shot	62.4 (61.0)	92.4 (91.6)	94.9 (93.3)	36.1 (35.2)	**10.8** (10.4)	29.4 (28.2)	29.4 (26.8)	11.0 (9.8)	22.3 (20.9)
	5-shot	62.6 (61.8)	93.1 (92.3)	94.5 (94.0)	35.8 (35.4)	10.6 (10.2)	29.5 (28.1)	30.7 (27.6)	11.3 (10.1)	22.8 (21.4)
UniICL♠	1-shot	59.1 (58.7)	93.0 (91.9)	94.5 (91.6)	34.8 (34.7)	10.4 (10.3)	28.1 (27.8)	29.1 (26.2)	10.8 (9.4)	22.2 (20.7)
	2-shot	62.6 (61.2)	94.0 (93.0)	94.9 (92.3)	34.6 (34.3)	10.6 (10.4)	28.5 (28.3)	30.3 (28.9)	11.3 (10.5)	22.9 (21.7)
	5-shot	63.3 (61.5)	**94.7** (92.8)	95.0 (93.8)	35.6 (35.3)	11.0 (10.8)	29.1 (27.7)	31.1 (30.0)	11.7 (11.2)	23.5 (22.3)
	8-shot	63.8 (62.6)	**94.7** (93.1)	95.0 (94.2)	-	-	-	-	-	-
UniICL♠ + $\mathcal{L}_{ctr}$	1-shot	59.3 (58.9)	93.2 (92.4)	95.1 (92.8)	35.6 (35.1)	10.7 (10.5)	28.9 (28.3)	30.0 (27.9)	11.3 (10.1)	22.8 (21.5)
	2-shot	62.4 (62.0)	94.5 (92.8)	94.8 (93.4)	36.8 (35.3)	10.8 (10.6)	29.6 (28.9)	30.8 (29.2)	11.4 (10.7)	23.0 (21.9)
	5-shot	64.3 (61.8)	**94.7** (93.4)	**96.1** (94.2)	**37.1** (34.9)	**11.3** (11.2)	**30.0** (29.3)	**32.5** (30.6)	**12.3** (11.8)	**24.7** (23.8)
	8-shot	**64.7** (63.3)	**94.7** (94.1)	95.6 (95.0)	-	-	-	-	-	-

Table 2: The high- and low-ICL results on CoLA-dev, SST-2-dev, and IMDb. Results in () represent low-resource ICL. ♠ represents the demonstrations selected by UniICL, and the others are selected by S-BERT. $+L_{ctr}$ indicates the selection augmented UniICL (optimized with Equation 7). **Bold** (underline) represents the best performance on high- and low-resource ICL.

Method	MS MARCO
BM25†	18.5
Vicuna	28.9
AutoCompressor	29.3
ICAE	30.2
UniICL	**31.6**

Table 3: Results on MS MARCO. Vicuna applies the last hidden states of [EOS] to represent sentences in latent space. Following the previous study, we report MRR@10. † means citing from Liang (Wang et al., 2022).

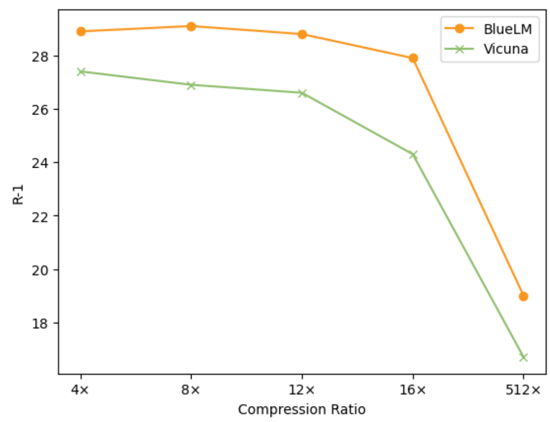

Figure 7: The overall sensitivity analysis of compression ratio.

Furthermore, The ablation experiments of $+\mathcal{L}_{ctr}$ show that UniICL is faced with performance degradation without L_{ctr} and the performance gap becomes larger with the number of demonstrations increasing. **The results of BlueLM are exhibited in Appendiex B**.

Passage Ranking Since the virtual tokens naturally summarize semantic information of preceding sequences, we evaluate UniICL on the out-domain MS MARCO dataset in Table 3. UniICL significantly outperforms the sparse retrieval method BM25 algorithm and other compression methods. Notably, we don't compare UniICL with the popular retrieval models (Reimers and Gurevych, 2019; Wang et al., 2024) since most of them are fine-tuned on this dataset, which is unfair for comparison.

5 Analysis

5.1 Compression Ratio

During training, the compression ratio is dynamically sampled from 2 to 16. We mix up 2,000 instances from the in-domain validation set, 1,000 for XSum, and 1,000 for CICERO to select the compression ratio for UniICL in Figure 7, with the backbone of Vicuna and BlueLM respectively. Specifically, UniICL compresses the latter cut-off part while keeping the former ones uncompressed.

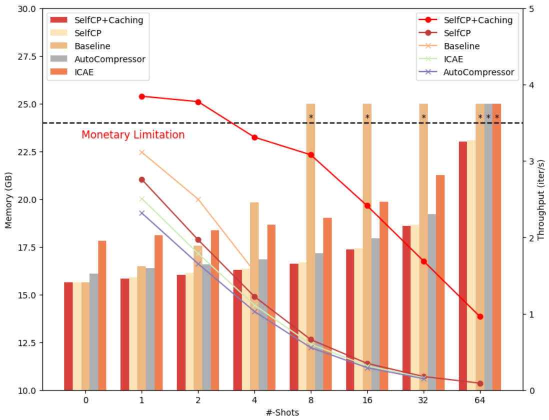

Figure 8: The efficiency comparison between UniICL and other compression methods in CoLA with the number of shots increasing from 0 to 64. Memory explodes are represented as *, corresponding to the break of the line chart.

Method	GPUHours	TFLOPs	TMACs
Vicuna	1.5	86,20	4,309
Vicuna-1k	1.9	31,664	15,832
UniICL	1.6	22,437	11,218

Table 4: The computation efficiency of UniICL.

Therefore, we can measure the dense information quality of the same content with different compression ratios by ROUGE-1 since it is more sensitive to token-level differences. The performance is relative smoothing when the compression ratio changes from $4\times$ to $12\times$. However, when it comes to $16\times$, an obvious drop occurs. Therefore, we set the compression ratio to 12 by default and apply this ratio to all experiments. The $512\times$ compression ratio is equal to compressing anything to a single virtual token, due to the maximum allowed input length for compression being 512.

5.2 Efficiency Analysis

In UniICL, we incorporate an additional 17M trainable parameters into the 7b backbone, accounting for an approximate increase of 0.24%. We evaluate the memory costs inference latency of UniICL and other compression methods in Figure 8. With the help of the Demonstration Bank (DB), UniICL will eliminate the extra latency if the selected demonstrations have been compressed and cached (UniICL+Caching). Despite this, parallel computation facilitates the compressing process, resulting in minimal throughput degradation (UniICL and Baseline). The naive 7B LLM occurs memory explosion for 8-shot settings and other compression methods perform up to 32-shot, while UniICL successfully scales up to 64-shots within 24GB CUDA allocation.

Additionally, We demonstrate the inference computation and GPU hours in Table 4, by using 1,024 random legal tokens as inputs and forcing models to generate 128 tokens. Notably, UniICL (without DB) compresses the former half, and the latter half is fed into the generator directly, while Vicuna and Vicuna-1k are distinguished in window limitations. Results indicate that minimal GPU hours increased due to the parallel computation of forward, although the extra compression of UniICL surges the computation. Additionally, Vicuna with a 1k window limitation surges both GPU hours and TFLOPs because long input brings significant computation and latency in generation.

5.3 Training Deviation

To quantify the performance gains brought by the learnable projection layer. We tune Vicuna and BlueLM with comparable parameters (17M) with LoRA in Table 5, setting the rank to 32. UniICL still outperforms LoRA-adapted LLMs with a 512 window limitation, indicating that the truncation indeed brings performance degradation.

6 Conclusion

This paper proposes UniICL, a parameter-efficient ICL framework that unifies demonstration selection, demonstration compression, and final response generation via a frozen LLM. Experimental results show that the generated virtual tokens substitute the $12\times$ longer actual demonstrations with minimal time expenditure, scaling up the number of demonstrations from 4 to 64.

7 Limitations

Our study, while proposing an efficient unified ICL framework for demonstration compression and selection, still has limitations. Firstly, UniICL is limited to the realm of naive ICL leaving other advanced LLM prompting methods, e.g. Retrieval Augment Generation (RAG) and Chain-of-Thought (CoT) unexplored. Limited to the hardware, we employ the underlying LLM at a scale of 7 billion parameters. Larger-scale LLMs are welcome to enrich our findings in future studies.

References

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2022. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Aydar Bulatov, Yuri Kuratov, and Mikhail S Burtsev. 2023. Scaling transformer to 1m tokens and beyond with rmt. *arXiv preprint arXiv:2304.11062*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*.

Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, and Furu Wei. 2023. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*.

Tao Ge, Jing Hu, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. In-context autoencoder for context compression in a large language model. *arXiv preprint arXiv:2307.06945*.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*.

Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer. 2022. Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 784–794.

Yucheng Li. 2023. Unlocking context constraints of llms: Enhancing context efficiency of llms with self-information-based content filtering. *arXiv preprint arXiv:2304.12102*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Noisy channel language model prompting for few-shot text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5316–5330.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. *arXiv preprint arXiv:2304.08467*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. *choice*, 2640:660.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *arXiv preprint arXiv:2302.00083*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Stanford alpaca: An instruction-following llama model.

BlueLM Team. 2023. Bluelm: An open multilingual 7b language model. https://github.com/vivo-ai-lab/BlueLM.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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.

Jiaan Wang, Yunlong Liang, Fandong Meng, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. 2023a. Is chatgpt a good nlg evaluator? a preliminary study. *arXiv preprint arXiv:2303.04048*.

Lean Wang, Lei Li, Damai Dai, Deli Chen, Hao Zhou, Fandong Meng, Jie Zhou, and Xu Sun. 2023b. Label words are anchors: An information flow perspective for understanding in-context learning. *arXiv preprint arXiv:2305.14160*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Simlm: Pre-training with representation bottleneck for dense passage retrieval. *arXiv preprint arXiv:2207.02578*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Large search model: Redefining search stack in the era of llms. In *ACM SIGIR Forum*, volume 57, pages 1–16. ACM New York, NY, USA.

Zengzhi Wang, Qiming Xie, Zixiang Ding, Yi Feng, and Rui Xia. 2023c. Is chatgpt a good sentiment analyzer? a preliminary study. *arXiv preprint arXiv:2304.04339*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2018. Neural network acceptability judgments. *arXiv preprint 1805.12471*.

Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, et al. 2023. Zero-shot information extraction via chatting with chatgpt. *arXiv preprint arXiv:2302.10205*.

David Wingate, Mohammad Shoeybi, and Taylor Sorensen. 2022. Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models. *arXiv preprint arXiv:2210.03162*.

Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. 2022. Memorizing transformers. *arXiv preprint arXiv:2203.08913*.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*.

Xianjun Yang, Yan Li, Xinlu Zhang, Haifeng Chen, and Wei Cheng. 2023. Exploring the limits of chatgpt for query or aspect-based text summarization. *arXiv preprint arXiv:2302.08081*.

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

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*.

Lin Zheng, Chong Wang, and Lingpeng Kong. 2022. Linear complexity randomized self-attention mechanism. In *International conference on machine learning*, pages 27011–27041. PMLR.

A In-Domain Evaluation

Backbone	Method	XSum			CICERO		
		R-1	R-2	R-L	R-1	R-2	R-L
Vicuna-7b	Vicuna	19.9	5.0	13.5	17.3	3.3	14.3
	+LoRA	25.4	7.5	17.3	28.1	10.5	25.6
	Vicuna-1k	27.3	8.7	19.7	30.5	11.3	27.4
	+LoRA	31.2	11.0	23.1	34.1	13.5	30.2
	UniICL	30.0	10.2	22.3	32.6	12.2	28.8
BlueLM-7b	BlueLM	15.0	3.6	10.4	17.6	3.1	15.0
	+LoRA	23.1	7.6	17.4	21.9	7.8	19.8
	BlueLM-1k	28.1	9.9	22.8	25.1	9.2	23.1
	+LoRA	30.8	10.5	24.6	31.2	10.8	27.4
	UniICL	30.4	10.2	23.7	29.2	10.0	26.6

Table 5: The in-domain results and ablation studies on XSum and CICERO. 1k represents the extending 1k window limitation, while others have a limitation of 512.

We conduct the zero-shot in-domain generation evaluation on the entire test set of XSum and CICERO in Table 5 by compressing the latter half to virtual tokens and keeping the former unmodified. UniICL significantly outperforms the baselines, indicating the compressed virtual tokens can provide the original truncated information by recovering the cut-off parts after supervised fine-tuning. Although extending the window to 1k, Vicuna and BlueLM still underperform UniICL, indicating that compressed virtual tokens filter noise information to some extent.

B Results on BlueLM

We extra conduct experiments on BlueLM (Team, 2023) to verify the generality of UniICL. We demonstrate the result of understanding tasks in Table 6, of the generative tasks in Table 7.

Model	#-shots	CoLA-dev	SST-2-dev	IMDb
			Acc.	
BlueLM	0-shot	71.6	81.2	48.8
	1-shot	69.6	82.6	64.8
	2-shot	70.0	87.0	65.6
	5-shot	70.5	88.6	68.7
UniICL	1-shot	69.6	81.2	65.4
	2-shot	68.7	82.6	67.0
	5-shot	71.7	87.0	70.4
UniICL♠	1-shot	69.8	80.0	62.0
	2-shot	70.1	80.8	67.0
	5-shot	71.8	85.6	69.6
	8-shot	72.3	87.4	69.4
UniICL♠ + L_{ctr}	1-shot	70.1	80	69.6
	2-shot	70.3	87.2	70.6
	5-shot	71.1	89.2	71.0
	8-shot	72.5	90.4	76.8

Table 6: The ICL results of understanding tasks with the backbone of BlueLM.

Method	#-shots	XSum			ARXIV		
		R-1	R-2	R-L	R-1	R-2	R-L
BlueLM	0-shot	15.0	3.6	10.4	30.9	7.7	24.7
	1-shot	19.1	4.8	12.1	23.0	3.6	19.0
UniICL	1-shot	24.0	6.9	18.0	31.4	7.7	25.2
	2-shot	25.0	7.3	18.8	30.8	7.3	24.8
	5-shot	25.3	7.4	19.1	31.9	7.8	26.0
UniICL♠	1-shot	25.2	7.4	18.9	31.6	7.9	25.4
	2-shot	25.4	7.6	19.1	31.9	8.0	25.6
	5-shot	26.5	7.9	20.3	32.1	8.0	25.5
UniICL♠ + L_{ctr}	1-shot	24.7	7.2	18.5	31.0	7.5	24.9
	2-shot	25.1	7.4	19.0	31.2	7.7	25.1
	5-shot	26.3	7.6	20.0	31.5	7.9	25.3

Table 7: The ICL results of generative tasks with the backbone of BlueLM.

Dataset	# words		
	(96,512]	(512,1024]	(1024,1536]
XSum	-	10,000	4,697
CICERO	10,000	-	-
SUPER-NI	-	10,000	7,000
XSum (Ctr)		5,000	

Table 8: The composition training set of UniICL. (m,n] represents the range of the number of words in each instance. XSum (Ctr) is used for the second phase training in Equation 7.

C Datasets & Metrics

C.1 Datasets

We mix up two public datasets for compression and selection augmentation training, described in Table 8. Additionally, UniICL achieves outstanding performance on out-domain evaluation, involving text summarization (Narayan et al., 2018), passage ranking (Nguyen et al., 2016), sentiment classification (Maas et al., 2011; Socher et al., 2013), and linguistic acceptability (Warstadt et al., 2018), more details referring to Table 9. UniICL selects demonstrations from its training set in high-resource ICL, and we fixed the number of candidate demonstrations to 20 for low-resource ICL evaluation.

Dataset	Task	In-Domain	# Test	# Demonstrations
MS MARCO-dev	Passage Ranking	✗	6,980	-
XSum	Text Summarization	✓	1,500	204,045/20
ARXIV	Text Summarization	✗	1,500	203,037/20
CoLA-dev	Lingustic Acceptability	✗	1,041	67,349/20
SST2-dev	Sentiment Classification	✗	872	8,551/20
IMDb	Sentiment Classification	✗	1,500	25,000/20

Table 9: The details of involved evaluation datasets. -dev represents employing development set due to their test sets are inaccessible. # Demonstrations represent the number of demonstrations to be selected in **high**/low-resource ICL settings.

C.2 Evaluation Metrics

ROUGE (Lin, 2004) is a widely adopted metric in many generative tasks that evaluate how similar the generated hypothesis is to the golden label. Therefore, ROUGE is used in our experiments to evaluate the quality responses generated conditioned on compressed virtual tokens, and we report the F-1 scores of ROUGE-1, ROUGE-2, and ROUGE-L (abbreviated R-1, R-2, R-L in the following), and we employed the files2rouge [8] library in practice. Following the previous works, we report the accuracy of close-ended evaluation and MRR@10 for passage ranking.

[8] https://github.com/pltrdy/files2rouge.