DisComp: A Two-Stage Prompt Optimization Framework Combining Task-Agnostic and Task-Aware Compression

Quancai Liu[1,2], **Haihui Fan**[1,2*], **Jinchao Zhang**[1,2], **Xiangfang Li**[1,2], **Chuanrong Li**[1,2], **Bo Li**[1,2]

[1]State Key Laboratory of Cyberspace Security Defense, Institute of Information Engineering, CAS, Beijing, China
[2]School of Cyber Security, University of the Chinese Academy of Sciences, Beijing, China
{liuquancai, fanhaihui, zhangjinzhao, lixiangfang, lichuanrong, libo}@iie.ac.cn

Abstract

Large language models (LLMs) exhibit exceptional performance across a wide range of natural language processing tasks, often relying on lengthy prompts to harness their full capabilities. However, extended prompts can lead to substantial computational overhead and increased hardware demands, limiting the scalability and efficiency of such models. In this paper, we propose **DisComp**, a two-stage prompt **comp**ression framework based on knowledge **dis**tillation that combines task-agnostic and task-aware strategies, designed to efficiently compress prompt length without compromising performance.

In the first stage, task-agnostic compression is achieved through knowledge distillation, transferring the summarization capabilities of a LLM to a smaller, more efficient model. The distillation process combines cross-entropy loss and keyword matching loss to ensure the smaller model generates concise and informative summaries. In the second stage, sentence-level pruning is applied, where sentences are ranked by relevance to the query, and irrelevant sentences are pruned to retain only task-critical information. We evaluate our method on three benchmark datasets, LongBench , ZeroSCROLLS and NaturalQuestions. The results show that DisComp significantly outperforms previous task-agnostic and task-specific compression approaches, and it is up to 6.56× faster at inference compared to the best token-level compression method.

1 Introduction

Large Language Models (LLMs) have continued to push the boundaries of natural language processing, achieving remarkable results in various tasks such as question answering, text summarization, and code generation (Ushio et al., 2023; Chowdhery et al., 2023). To maximize the potential of LLMs, researchers have introduced several advanced prompting techniques, including In-Context Learning (ICL) (Dong et al., 2023), Chain-of-Thought (CoT) reasoning (Wei et al., 2022), and Retrieval-Augmented Generation (RAG) (Lewis et al., 2020). While these methods can elicit highly effective generations by activating the domain-specific knowledge of LLMs, they often require longer prompts. However, the efficiency and performance of LLMs deteriorate when handling long input sequences due to the quadratic complexity of the self-attention mechanism inherent in Transformer architectures (Vaswani et al., 2017). Therefore, effectively compressing context length while maintaining model performance has emerged as a critical research challenge.

Recent research has introduced various methods for prompt compression to enhance computational efficiency when handling long contexts, while minimizing the impact on model performance. In task-aware compression, studies focus on customizing prompts based on specific tasks or queries, which has been shown to improve performance in downstream applications (Jiang et al., 2023b; Xu et al., 2024; Jung and Kim, 2023). However, task-aware compression often relies on task-specific features, limiting its efficiency and generalizability, particularly in multi-task or open-domain settings (Huang et al., 2023). In contrast, task-agnostic compression methods aim for broader applicability by removing redundant information (Li et al., 2023; Jiang et al., 2023a; Pan et al., 2024). These approaches utilize information entropy to eliminate unnecessary lexical units, offering greater generalization and efficiency. Nonetheless, the main challenges of task-agnostic methods lie in the imprecision of information entropy as a compression criterion and the inability of causal language models to fully capture essential contextual information due to their unidirectional nature.

To address the aforementioned issues, we propose a novel method that integrates task-agnostic and task-aware strategies, as shown in Figure 1. In the first stage, we apply task-agnostic compression based on knowledge distillation to transfer the summarization capabilities of a large language model (LLM) to a small language model (SLM). The distillation loss includes two components: cross-entropy loss, which aligns the small model's output with that of the LLM, and keyword matching loss, which ensures essential information is retained. This dual-loss approach enables the small model to generate concise and semantically rich summaries. In the second stage, we perform sentence-level pruning on the summaries generated by the small model. Sentences are ranked based on their relevance to the given query using a designed scoring function (e.g., using cosine similarity of contextual embeddings), and those with lower scores are pruned according to the compression ratio. This ensures that only the most relevant information is retained. By combining task-agnostic and task-aware strategies, our approach strikes an effective balance between efficiency and generalization.

Generally, our two-stage approach overcomes the limitations of solely relying on task-aware or task-agnostic methods, achieving comprehensive efficiency improvements. Our main contributions are as follows:

- **Unified Task-Agnostic and Task-Aware Compression:** To our knowledge, we are the first to integrate both task-agnostic and task-aware prompt compression strategies into a single cohesive framework. This unified approach leverages the strengths of each method, enabling efficient reduction of prompt length while ensuring the retention of task-specific critical information.

- **Two-Stage Compression Framework:** Our approach employs knowledge distillation and sentence-level pruning, generating concise, semantically complete summaries that significantly improve computational efficiency and model performance without compromising relevance to the specific task.

- **Accelerate Inference:** We evaluated the effectiveness of the proposed method on the LongBench (Bai et al., 2023), ZeroSCROLLS (Shaham et al., 2023) and NaturalQuestions (Liu et al., 2023a) datasets to assess its performance across various tasks. Our experimental results demonstrate that DisComp outperforms prior compression works and is up to 6.56x faster during inference compared to the existing best method.

2 Related Work

2.1 Knowledge Distillation

Recent advancements in knowledge distillation have introduced significant techniques, such as symbolic knowledge distillation (West et al., 2022). This approach transfers knowledge from a teacher model by generating a training dataset through the teacher model and subsequently training a student model on this dataset. To enhance the effectiveness of the distillation process, a critic criterion is introduced to filter out undesirable examples from the generated dataset, ensuring that the student model learns high-quality knowledge. This distillation technique has been widely applied across various tasks, including summarization (Jung et al., 2023), where the goal is to generate high-quality summaries while optimizing the performance of downstream language models (LMs). One work that is similar to our setting is (Hsu and Tan, 2021) which trains an extractive summarization model to optimize for prediction accuracy of a sentiment prediction model based on the summary.

2.2 Prompt Compression

Depending on whether task information is used for compression, prompt compression methods can be categorized into task-aware and task-agnostic compression approaches. Task-aware compression tailors the context to specific downstream tasks or queries. For example, LongLLMLingua (Jiang et al., 2023b) employs a question-aware coarse-to-fine strategy that estimates token information entropy based on the question. Reinforcement Learning (RL) methods (Jung and Kim, 2023; Huang et al., 2023) train models using reward signals from downstream tasks, while soft prompt tuning techniques (Wingate et al., 2022; Mu et al., 2023) require fine-tuning for specific tasks. Additionally, Xu et al. (Xu et al., 2024) utilize a summarization model to compress context conditioned on the question. Although task-aware methods achieve effective compression for particular tasks and ratios, their adaptability to diverse real-world applications is limited. In contrast, task-agnostic compression methods do not consider specific tasks, enhancing their applicability across various applications and compatibility with black-box large language mod-

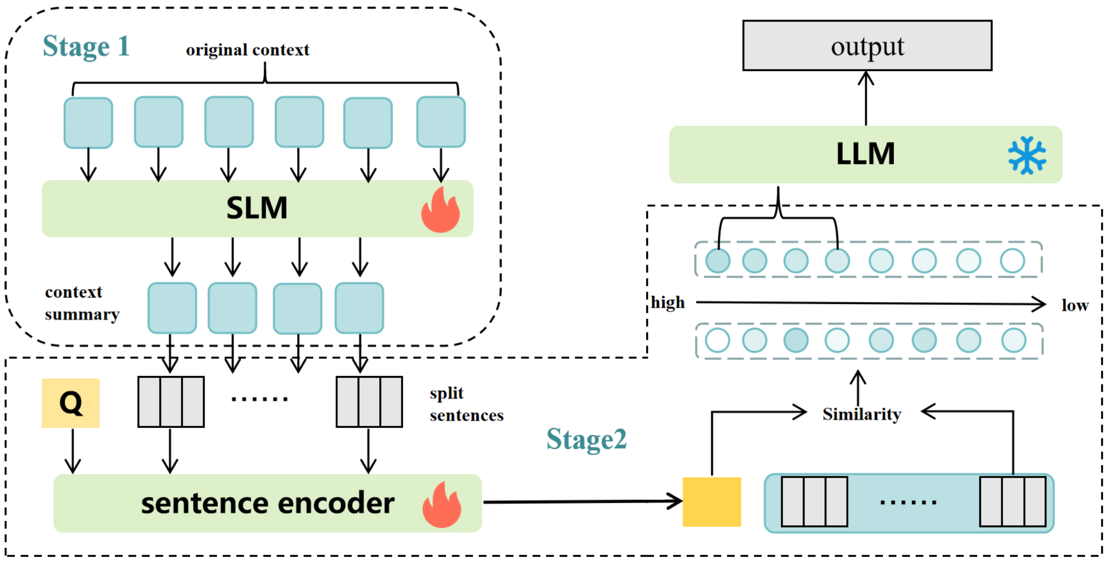

Figure 1: Overview of DisComp. DisComp utilizes a trained summarization model (SLM) and a sentence encoder to perform summarization and sentence-level pruning on the original context. The remaining sentences after pruning are then passed to the downstream LLM for output generation.

els (LLMs). These methods typically use information entropy metrics to eliminate redundant prompt information (Li et al., 2023; Jiang et al., 2023a) or employ summarization techniques (Chen et al., 2023; Packer et al., 2023). However, they may fail to optimally capture token importance for specific LLMs and often involve high computational costs, while summarization-based approaches can omit essential details. Additionally, methods that compress hidden or KV caches (Chevalier et al., 2023; Ge et al., 2023; Liu et al., 2023b; Xiao et al., 2024) are not easily applicable to black-box LLMs and are thus beyond the scope of this work.

3 Method

3.1 Problem Definition

A prompt compression system is designed to generate a compressed prompt $\tilde{x} = \{\tilde{x}_i\}_{i=1}^{\tilde{L}}$ from an input context $x = \{x_i\}_{i=1}^{L}$, where $\tilde{L}$, L represent the numbers of tokens in $\tilde{x}$, x, respectively, and $\tilde{L} < L$. The compression ratio can be denoted as $\tau = \tilde{L}/L$, where $\tau \in [0, 1]$. DISCOMP comprises two sequential stages: task-agnostic summarization and task-aware sentence-level pruning. In the first stage, a task-agnostic compressor condenses the input context x into a summary s, effectively capturing the core information of x while significantly reducing the number of tokens. In the second stage, the summary s is segmented into individual sentences $\{S_i\}_{i=1}^{K}$, where K is the number of the sentences. Each sentence is processed in conjunction with the query Q by a task-aware encoder, and sentences that receive scores below a predetermined threshold are subsequently pruned.

3.2 Task-agnostic Summarization

In our process of distilling the capabilities of a large teacher model into a smaller student model, we adopt a task-agnostic summarization approach. Unlike previous methods that guide summarization using input queries or task-specific prompts, we provide a handcraft summarization prompt, which directs the teacher model to summarize the given context C_i without referencing any specific input or task-specific queries Q. This strategy enables the model to learn general summarization abilities without reliance on task-specific inputs. The detailed procedure is outlined in Algorithm 1.

Our training dataset is constructed from the Wikitext-103 dataset (Merity et al., 2017). During data construction, we extract continuous text passages from Wikitext-103 to form a set of contexts $\{C_i\}_{i=1}^{N}$, where N is the number of samples. For each context C_i, we use a handcraft prompt to guide the teacher model in generating the corresponding summary S_i. The generated summaries encapsulate the main information within the con-

Algorithm 1 Fine-tuning Student Model M_s

```
Input: Teacher model $M_t$, Student model $M_s$, Set of contexts $\{C_i\}_{i=1}^{N}$, Handcrafted summarization prompt $p$, hyperparameter $\lambda$
Output: Fine-tuned student model $M_s$
 1: $D \leftarrow \emptyset$
 2: for $i \in \{1, 2, \ldots, N\}$ do
 3:     $S_i = \text{Generate}(M_t, [p; C_i])$
 4:     $K_i = \text{ExtractKeywords}(S_i)$
 5:     $D \leftarrow D \cup \{(C_i, S_i, K_i)\}$
 6: end for
 7: $L_{\text{total}} \leftarrow 0$
 8: for $(C_i, S_i, K_i) \in D$ do
 9:     $\hat{S}_i = \text{Generate}(M_s, C_i)$
10:     $L_{\text{CE},i} = -\sum_{t=1}^{T_i} \log P_S(S_i^t \mid S_i^{<t}, C_i)$
11:     $L_{\text{KW},i} = 1 - \frac{|K_i \cap \hat{S}_i|}{|K_i|}$
12:     $L_i = L_{\text{CE},i} + \lambda L_{\text{KW},i}$
13:     $L_{\text{total}} \leftarrow L_{\text{total}} + L_i$
14: end for
15: $M_s \leftarrow \text{Update}(M_s, \nabla_{M_s} L_{\text{total}})$
```

texts, allowing the model to focus on concise content expression.

When training the student model, our goal is to ensure that the student model replicates the summaries produced by the teacher model. To achieve this, we design a combined loss function that integrates cross-entropy loss and keyword matching loss. The cross-entropy loss L_{CE} measures the token-level differences between the student's generated summary $\hat{S}_i$ and the teacher's summary S_i:

$$L_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T_i} \log P_{\text{S}}\left(S_i^t \mid S_i^{<t}, C_i\right)$$

Here, T_i is the length of the summary S_i, and P_{S} represents the probability distribution of the student model.

The keyword matching loss L_{KW} ensures that the student's generated summary retains the key information from the teacher's summary. We extract a set of keywords K_i from the teacher's summary S_i using the RAKE algorithm (Rapid Automatic Keyword Extraction). RAKE is an unsupervised, domain-independent keyword extraction algorithm that identifies key phrases in a text by analyzing the frequency of word appearances and their co-occurrences with other words. The extracted keywords are then used in the loss calculation:

$$K_i = \text{ExtractKeywords}(S_i) \tag{2}$$

The keyword matching loss is defined as the proportion of keywords not included in the student's summary:

$$L_{\text{KW}} = \frac{1}{N}\sum_{i=1}^{N}\left(1 - \frac{\left|K_i \cap \hat{S}_i\right|}{|K_i|}\right) \tag{3}$$

The total loss function is:

$$L_{\text{total}} = L_{\text{CE}} + \lambda L_{\text{KW}} \tag{4}$$

where λ is a hyperparameter used to balance the importance of textual similarity and content preservation.

3.3 Task-aware Sentence-level Pruning

We propose a novel method to train a sentence encoder model by leveraging the summaries generated by our fine-tuned student model and the question-answer (QA) pairs generated by prompting a downstream LLM. The detailed procedure is outlined in Algorithm 2. Specifically, for each context C_i, we perform the following steps:

Summary Generation: We use the model M_s obtained from the previous step to generate a summary S_i of the context C_i. The summary S_i is then segmented into individual sentences $\{s_{ij}\}_{j=1}^{N_i}$, where N_i is the number of sentences in S_i. **QA Pair Generation**: To prepare for subsequent task-aware sentence-level pruning, we prompt a frozen downstream LLM to generate corresponding QA pairs $\{(Q_{ik}, A_{ik})\}_{k=1}^{M_i}$ for each context C_i, where M_i denotes the number of QA pairs generated from C_i. In these generated QA pairs, A_{ik} serves as the reference answer. **Sentence-Question Scoring**: The encoder model enc_θ embeds each sentence s_{ij} and question Q_{ik} into fixed-dimensional embeddings. The inner product of these embeddings represents how helpful the sentence s_{ij} is for answering the question Q_{ik}. We compute a score for each sentence-question pair using a combined function that considers both the likelihood of generating the correct answer and the semantic similarity between the sentence and the question:

$$\begin{aligned}\text{Score}(s_{ij}, Q_{ik}) &= \alpha \cdot \log p_M(A_{ik} \mid [s_{ij}; Q_{ik}]) \\ &\quad + (1-\alpha) \cdot \text{sim}\left(\text{enc}_\theta(s_{ij}),\ \text{enc}_\theta(Q_{ik})\right)\end{aligned} \tag{5}$$

where $\alpha \in [0, 1]$ balances the two components, p_M is the probability assigned by a language model M, and $\text{sim}(\cdot, \cdot)$ is the cosine similarity between embeddings.

Algorithm 2 Training Sentence Encoder Model enc_θ

Input: Fine-tuned student model M_s, Frozen LLM M, Set of contexts $\{C_i\}_{i=1}^N$, Hyperparameters β, τ, P

Output: Trained sentence encoder model enc_θ

```
1:  $D \leftarrow \emptyset$
2:  for $i \in \{1, 2, \ldots, N\}$ do
3:      $S_i = \text{Generate}(M_s, C_i)$
4:      $\{s_{ij}\} = \text{Segment}(S_i)$
5:      $\{(Q_{ik}, A_{ik})\} = \text{GenerateQA}(M, C_i)$
6:      for $(s_{ij}, Q_{ik})$ do
7:          $score(s_{ij}, Q_{ik}) = \beta \cdot \log p_M(A_{ik} \mid [s_{ij}; Q_{ik}]) + (1 - \beta) \cdot \text{sim}(enc_\theta(s_{ij}), enc_\theta(Q_{ik}))$
8:      end for
9:      $P_i \leftarrow$ Top-K sentences by score
10:     $N_i \leftarrow \text{HardNegativeSampling}(P_i)$
11:     $D \leftarrow D \cup \{(P_i, N_i, Q_{ik})\}$
12: end for
13: $L = -\log\left(\frac{\sum \text{pos}}{\sum \text{pos} + \sum \text{neg}}\right)$
14: Update $enc_\theta$ with $\nabla_\theta L$
```

Selection of Positive and Negative Examples: We select the top P sentences with the highest scores as positive examples P_i. Negative examples N_i are selected through hard negative sampling by choosing sentences that are semantically similar to the positive examples but have lower scores. **Model Training**: We train a encoder model enc$_\theta$ that embeds sentences s_{ij} and questions Q_{ik} into fixed-dimensional embeddings. The model is trained using the NT-Xent (Normalized Temperature-scaled Cross Entropy) loss function:

$$L = -\log \frac{\sum_{p_j \in P_i} \textbf{pos}}{\sum_{p_j \in P_i} \textbf{pos} + \sum_{n_j \in N_i} \textbf{neg}} \quad (6)$$

where,

$$\textbf{pos} = \exp\left(\text{sim}(\text{enc}_\theta(p_i), \text{enc}_\theta(Q_{ik}))/\tau\right) \quad (7)$$

$$\textbf{neg} = \exp\left(\text{sim}(\text{enc}_\theta(n_i), \text{enc}_\theta(Q_{ik}))/\tau\right) \quad (8)$$

τ is a temperature parameter.

3.4 Inference

At the inference stage, given an input context x and a query Q, we repeat the three steps outlined previously: summary generation, sentence-question scoring, and selection of positive and negative examples. First, the input context is processed through a summarization model, compressing it to τ_1 times its original length, where τ_1 represents the summary compression ratio. Then we obtain the summarized context x_1, which is then divided into a set of sentences $\{s_i\}$.

Next, sentences are scored; lower-scoring sentences are pruned, and higher-scoring sentences are retained. The number of retained sentences is determined based on the pruning compression ratio τ_2. Specifically, we select the top K sentences with the highest scores such that their total token length satisfies $\sum_{i=1}^{K} \text{tokenlen}(S_i) \leq \tau_2 \cdot \text{tokenlen}(x_1)$.

Subsequently, the selected sentences are reordered according to their original sequence in the context to form the compressed context $\tilde{x}$. The final compression ratio for the entire process is given by $\tau = \tau_1 \cdot \tau_2$. Finally, the compressed context $\tilde{x}$ is fed into the downstream language model to generate the output.

4 Experiments

4.1 Dadasets

To evaluate the effectiveness of our method, we follow the experimental setup of previous works (Jiang et al. 2023c), we use the three benchmark datasets: LongBench (Bai et al., 2023), ZeroSCROLLS (Shaham et al., 2023) and NaturalQuestions (Liu et al., 2023a). LongBench is a dataset designed for long-context scenarios and contains six task type tasks, such as single-document question answering (QA) and multi-document QA. Similarly, ZeroSCROLLS is a multi-task dataset consists of four task types. NaturalQuestions is similar to a retrieval-augmented generation benchmark, where each question is paired with 20 documents, one of which contains the correct answer, and accuracy is used as the evaluation metric.

4.2 Implementation Details

For task-agnostic summarization, we employ a student model with an encoder-decoder architecture(775M), initialized from the T5-large checkpoint (Raffel et al., 2020). This model has been trained with summarization datasets (Hermann et al., 2015). We fine-tune it on the constructed dataset using the combined loss function. The teacher model used for summarization is GPT-3.5-Turbo, which generates high-quality summaries for training the student model. To ensure stable and reproducible results, we employ greedy decoding

Methods	LongBench									ZeroSCROLLS		
	SingleDoc	MultiDoc	Summ.	FewShot	Synth.	Code	AVG	Tokens	$1/\tau$	AVG	Tokens	$1/\tau$
2,000 tokens constraint												
Task-Agnostic Compression												
Selective-Context	16.2	34.8	24.4	15.7	8.4	49.2	24.8	1,865	5x	20.5	1,773	6x
LLMLingua	22.4	32.1	24.5	61.2	10.4	56.8	34.6	1,862	5x	27.2	1,784	5x
LLMLingua-2	29.8	33.1	25.3	66.4	21.3	**58.9**	39.1	1,898	5x	33.3	1,862	5x
Task-Aware Compression												
SBERT	33.8	35.9	25.9	23.5	18.0	17.8	25.8	1,947	5x	20.5	1,773	6x
OpenAI	34.3	36.3	24.7	32.4	26.3	24.8	29.8	1,991	5x	24.0	1,784	5x
LongLLMLingua	39.0	42.2	27.4	69.3	**53.8**	56.6	48.0	1,809	6x	32.5	1,753	6x
DisComp(Ours)	**42.3**	**46.1**	**29.2**	**70.2**	51.5	58.3	**49.6**	1,796	6x	**33.4**	1,898	5x
3,000 tokens constraint												
Task-Agnostic Compression												
Selective-Context	23.3	39.2	25.0	23.8	27.5	53.1	32.0	3,417	3x	19.8	1,865	5x
LLMLingua	31.8	37.5	26.2	67.2	8.3	53.2	37.4	3,399	3x	24.0	1,862	5x
LLMLingua-2	35.5	38.7	26.3	69.6	21.4	**62.8**	42.4	3,421	3x	29.3	1,898	5x
Task-Aware Compression												
SBERT	35.3	37.4	26.7	63.4	51.0	34.5	41.4	3,328	3x	20.7	1,773	6x
OpenAI	34.5	38.6	26.8	63.4	49.6	37.6	41.7	3,421	3x	24.0	1,784	5x
LongLLMLingua	40.7	46.2	27.2	70.6	**53.0**	55.2	48.8	3,283	3x	32.8	1,753	6x
DisComp(Ours)	**42.5**	**46.7**	**29.5**	**71.4**	52.0	59.7	**50.3**	3,352	3x	**34.9**	3,327	3x
Original Prompt	39.7	38.7	26.5	67.0	37.8	54.2	44.0	10,295	-	34.7	9,788	-
Zero-Shot	15.6	31.3	15.6	40.7	1.6	36.2	23.5	214	48x	10.8	32	306x

Table 1: Performance of different models across various compression ratios on LongBench and ZeroSCROLLS.

and set the temperature to 0 in all experiments. **For task-aware sentence-level pruing**, we adopt Mistral-7B-Instruct-v0.2 (Jiang et al., 2023c) as our sentence encoder and fine-tune this model with LoRA (Hu et al., 2022) of rank 16 to learn the context-aware embeddings.

We alo use GPT-3.5-Turbo as the downstream targhet LLM. We fine-tune the two aforementioned models using the AdamW optimizer, with learning rates of 3e-5 and 5e-5, and a batch size of 32. Our method is implemented using Huggingface's Transformers library and PyTorch 2.0.1, with training conducted on a single NVIDIA A100 80G GPU using CUDA-12.1 for hardware.

4.3 Baselines

We compare our method with several state-of-the-art prompt compression techniques. As primary baselines, we adopt **Selective-Context** (Li et al., 2023), **LLMLingua** (Jiang et al., 2023a), and **LLMLingua-2** (Pan et al., 2024), which represent the latest advancements in task-agnostic prompt compression. Additionally, we include task-aware prompt compression methods in our evaluation, including retrieval-based methods: BM25, Gzip (Jiang et al., 2023b), Sentence-BERT (Reimers and Gurevych, 2019), OpenAI's text-embedding-ada-002 model, and **LongLLMLingua**(Jiang et al., 2023b). designed for handling long-context scenarios. By incorporating both task-agnostic and task-aware methods, we provide a comprehensive comparison of different prompt compression approaches across various tasks and contexts.

4.4 Results

Main results. We evaluated the effectiveness of the proposed method on the LongBench, ZeroSCROLLS, and NaturalQuestions datasets to assess its performance across various tasks. Our approach was compared against multiple baseline methods, including both task-agnostic and task-aware compression techniques.

Table 1 presents the main results of our method on the LongBench and ZeroSCROLLS datasets, evaluated under token constraints of 2,000 and 3,000 tokens, corresponding to approximately 5× and 3× compression rates. We assessed performance across multiple subtasks, including single-

Methods	GPT3.5-Turbo					LongChat-13b					Length	
	1st	5th	10th	15th	20th	1st	5th	10th	15th	20th	Tokens	$1/\tau$
2x constraint												
BM25	53.7	49.3	47.9	49.9	46.9	50.9	44.9	44.1	42.9	43.2	1545	1.9x
Gzip	64.6	63.8	60.5	58.3	57.3	61.9	55.7	52.7	50.8	50.9	1567	1.9x
SBERT	72.5	67.9	63.3	65.0	66.2	65.8	57.5	54.9	53.4	55.7	1549	1.9x
OpenAI	73.0	65.6	65.5	65.4	65.5	65.9	57.5	56.2	54.2	55.7	1550	1.9x
LongLLMLingua	77.2	72.9	70.8	70.5	70.6	68.7	59.4	57.3	55.9	58.4	1429	2.1x
DisComp(Ours)	**77.8**	**73.5**	**71.6**	**71.2**	**70.8**	**69.3**	**60.1**	**58.7**	**56.8**	**58.9**	1408	2.1x
4x constraint												
BM25	40.6	38.6	38.2	37.4	36.6	39.5	37.5	36.8	36.4	35.5	798	3.7x
Gzip	63.1	61.0	59.8	61.1	60.1	57.6	52.9	51.0	50.1	50.4	824	3.6x
SBERT	66.9	61.1	59.0	61.2	60.4	62.6	56.6	55.1	53.9	55.0	808	3.6x
OpenAI	63.8	64.6	65.4	64.1	63.7	61.2	56.0	55.1	54.4	55.0	804	3.7x
LongLLMLingua	75.0	71.8	71.2	71.2	74.7	68.7	60.5	59.3	58.3	61.3	748	3.9x
DisComp(Ours)	**75.6**	**72.3**	**72.0**	**71.6**	**75.0**	**69.8**	**61.2**	**60.0**	**58.7**	**62.1**	794	3.7x
Original Prompt	75.7	57.3	54.1	55.4	63.1	68.6	57.4	55.3	52.5	55.0	2946	-
Zero-shot			56.1					35.0			15	196x

Table 2: Results on NaturalQuestions with different compression ratio in a task-aware setting.

document comprehension, multi-document analysis, summarization, few-shot learning, synthetic tasks, and code understanding. On the LongBench dataset, except for slightly lower performance in the synthetic tasks and code understanding, our method outperforms the state-of-the-art methods in all other subtasks. Additionally, with 2,000 and 3,000 tokens constraints, the average performance improved by 1.6 and 1.5 percentage points, respectively. Similarly, on the ZeroSCROLLS dataset, our method surpasses the current state-of-the-art by 0.6 and 1.2 percentage points with 2,000 and 3,000 tokens constraints, respectively, while using the fewest tokens. This consistent superior performance across different datasets and compression rates further validates the strong generalization capability and broad applicability of our approach.

Table 2 presents the main results on the NaturalQuestions dataset. It can be observed that regardless of the position of the correct answer within the prompt, DisComp consistently achieves the best performance. Compared to the original prompt, DisComp utilizes fewer tokens while delivering superior results. For instance, under the 2x constraint, DisComp gains a performance improvement of 17.9% on GPT-3.5 Turbo with the ground-truth document at the 10th position. Similarly, under the 4x constraint, DisComp achieves a 7.1% performance boost on LongChat-13b with the ground-truth document at the 20th position. We can also observe that the position of the correct answer within the prompt significantly impacts the accuracy of the results, as shown in Figure 2. Accuracy is highest when the answer is placed in the first position, and it remains the second highest when placed in the 20th position, which is the last. Between these two extremes, accuracy decreases progressively. This suggests that large language models are more sensitive to the information presented at the beginning and the end of the input prompt, while paying less attention to the information in the middle.

Ablation studies. We conduct ablation studies to assess the contribution of different components of our proposed method. These experiments are performed on the LongBench Single-Doc subset, comprising NarrativeQA, Qasper, and MultiFieldQaen datasets, under a 2,000-token constraint. We evaluate the impact of removing key components, including keyword matching loss, sentence pruning, task-agnostic compression, knowledge distillation, and the sentence scoring mechanism during pruning. The accuracy scores reported in Table 3 illustrate the degradation in performance when

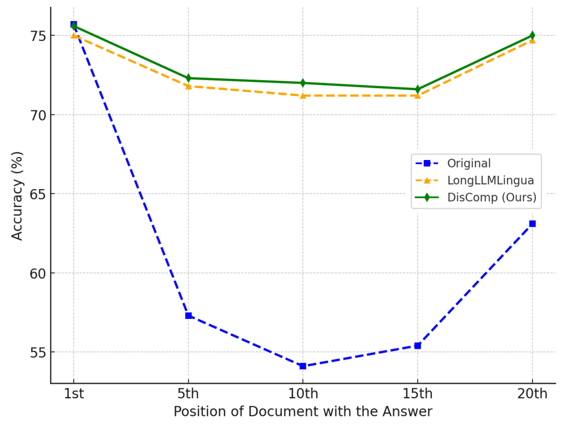

Figure 2: Performance Variation with Key Information in Different Positions.

each component is removed from the full model.

Removing the keyword matching loss impairs the model's ability to prioritize essential keywords, resulting in a noticeable decline in performance across datasets. The exclusion of sentence pruning leads to the inclusion of redundant information, negatively affecting model efficacy. Without task-agnostic compression, which utilizes knowledge distillation to produce efficient summaries, summary quality suffers, causing lower accuracy. Eliminating knowledge distillation hampers the student's ability to generalize effectively from the teacher model, leading to a substantial performance drop. Finally, without the sentence scoring mechanism—where sentences are selected randomly instead of based on relevance—the model exhibits a sharp decline in performance, underscoring the importance of this component in retaining relevant information in the compressed context.

Setup	**Accuracy**
DisComp	**42.15**
w/o L_{KW}	42.02
w/o Task-Agnostic Summarization	41.83
w/o Sentence Pruning	38.64
w/o Knowledge Distillation	40.34
w/o Score(S, Q)	39.46

Table 3: Ablation Study on LongBench Single-Doc Subset (Accuracy as the Sole Metric, 2,000-token Constraint)

Latency Evaluation. In this section, we evaluate the latency of our proposed method and compare it with prior state-of-the-art approaches in prompt compression. The results are summarized in Table 4, with all evaluations performed on an A100 GPU, the same hardware setup used for training. We measure the average processing time of samples from the LongBench dataset to ensure a fair comparison across methods. Specifically, we compare our approach against three recent methods: LLMLingua (Jiang et al., 2023a), LLMLingua-2 (Pan et al., 2024), and LongLLMLingua (Jiang et al., 2023b).

As shown in Table 4, DisComp significantly outperforms LLMLingua and LongLLMLingua in terms of latency, achieving a **13.88×** and **22.65×** speedup in average processing time, and a **5.9×** and **6.56×** speedup in median processing time, respectively. Although LLMLingua-2 has slightly lower latency than DisComp, it performs significantly worse across multiple benchmark tasks. In certain specific tasks, LLMLingua-2 even underperforms LongLLMLingua. Unlike previous methods, DisComp not only maintains competitive latency but also consistently achieves better performance in downstream tasks, providing a good balance between efficiency and effectiveness.

Method	**Avg.**	**Med.**	**Avg. Rel.**	**Med. Rel.**
LLMLingua	4.73	1.47	13.88×	5.9×
LLMLingua-2	0.23	0.10	0.68×	0.4×
LongLLMLingua	7.70	1.64	22.65×	6.56×
DisComp(Ours)	0.34	0.25	1×	1×

Table 4: Latency comparison across different methods. Avg., Med., and Rel. denote average, median, and relative times.

5 Conclusion

Our proposed method effectively integrates task-agnostic and task-aware compression strategies to maximize the efficiency and performance of large language models. By combining knowledge distillation with sentence-level pruning, we achieve a balance between generalization and task-specific optimization, enabling significant prompt compression while retaining crucial information. Experimental results demonstrate that our approach consistently outperforms existing methods across various tasks, validating its ability to address the challenges associated with long input sequences in LLMs. This work presents a promising direction for enhancing the efficiency of LLMs without sacrificing performance, contributing to the advancement of prompt optimization techniques.

Limitations

DisComp has limitations with respect to the training dataset. The training data used in this study is primarily based on Wikitext-103, which mainly focuses on encyclopedia-style text. While this dataset covers a wide range of topics, the model's generalization capability might be limited when dealing with other types of text. As demonstrated in the experiments on the LongBench dataset, although our method achieved the best performance in tasks such as single-document and multi-document question answering, summarization, and few-shot learning, it performed worse than the current state-of-the-art methods in synthetic and code understanding tasks. This dataset bias limits DisComp's adaptability in more diverse task scenarios, and further exploration is needed to assess its effectiveness across different domain data.

Additionally, DisComp shows certain limitations under high compression ratios. As the compression ratio increases, the model inevitably loses some critical information while trying to reduce the context length. Although task-agnostic compression strategies help preserve the core content, when the compression ratio is too high—particularly when handling long texts and complex contexts—the model's semantic integrity and information accuracy may be compromised. This can significantly degrade performance in tasks that require precise information retention. Therefore, future work could further optimize the compression algorithm to reduce information loss while improving efficiency.

Acknowledgments

This work was supported by the project No. ZTZB-23-990-024.

References

Yuzhong Bai, Xin Lv, Jipeng Zhang, Haobo Lyu, Jian Tang, Zhe Huang, Zhen Du, Xiangru Liu, Aowen Zeng, Liang Hou, et al. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Howard Chen, Ramakanth Pasunuru, Jason Weston, and Asli Celikyilmaz. 2023. Walking down the memory maze: Beyond context limit through interactive reading. *arXiv preprint arXiv:2310.05029*.

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

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2023. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

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*.

Karl Moritz Hermann, Tomáš Kočiskỳ, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NeurIPS)*, pages 1693–1701.

Chao-Chun Hsu and Chenhao Tan. 2021. Decision-focused summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 117–132, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *Proceedings of the 2022 International Conference on Learning Representations (ICLR)*.

Xijie Huang, Li Lyna Zhang, Kwang-Ting Cheng, and Mao Yang. 2023. Boosting llm reasoning: Push the limits of few-shot learning with reinforced in-context pruning. *arXiv preprint arXiv:2312.08901*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023a. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 13358–13376, Singapore. Association for Computational Linguistics.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023b. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *arXiv preprint arXiv:2310.06839*.

Xiang Jiang et al. 2023c. Mistral-7b-instruct-v0.2: An efficient large language model for instruction following. *arXiv preprint arXiv:2309.00001*.

Hoyoun Jung and Kyung-Joong Kim. 2023. Discrete prompt compression with reinforcement learning. *arXiv preprint arXiv:2308.08758*.

Jaehun Jung, Peter West, Liwei Jiang, Faeze Brahman, Ximing Lu, Jillian Fisher, Taylor Sorensen, and Yejin Choi. 2023. Impossible distillation: From low-quality model to high-quality dataset & model for summarization and paraphrasing. *arXiv preprint arXiv:2305.16635*.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6342–6353, Singapore. Association for Computational Linguistics.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023a. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2023b. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. In *Proceedings of the Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proceedings of the 2017 International Conference on Learning Representations (ICLR)*.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. Learning to compress prompts with gist tokens. In *Proceedings of the Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*.

Charles Packer, Vivian Fang, Shishir G Patil, Kevin Lin, Sarah Wooders, and Joseph E Gonzalez. 2023. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Ruhle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. *arXiv preprint arXiv:2403.12968*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):5485–5551.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Uri Shaham, Maor Ivgi, Avia Efrat, Jonathan Berant, and Omer Levy. 2023. Zeroscrolls: A zero-shot benchmark for long text understanding. *arXiv preprint arXiv:2305.14196*.

Asahi Ushio, Fernando Alva-Manchego, and Jose Camacho-Collados. 2023. An empirical comparison of lm-based question and answer generation methods. *arXiv preprint arXiv:2305.17002*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Peter West, Chandra Bhagavatula, Jack Hessel, Jena Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. 2022. Symbolic knowledge distillation: From general language models to commonsense models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4602–4625, Seattle, United States. Association for Computational Linguistics.

David Wingate, Mohammad Shoeybi, and Taylor Sorensen. 2022. Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5621–5634, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*.

A Keyword Extraction Algorithm

Algorithm 3 Keyword Extraction Algorithm

```
Input: Text summary $S_i$
Output: Keyword set $K_i$
 1: Define stopword list $StopWords$
 2: Use $StopWords$ and punctuation to split $S_i$ into a set of candidate phrases $P$
 3: Initialize frequency table $freq$ and degree table $deg$
 4: for each candidate phrase $phrase \in P$ do
 5:     Get the set of words in the phrase, denoted as $words$
 6:     for each word $w \in words$ do
 7:         $freq[w] \mathrel{+}= 1$
 8:         $deg[w] \mathrel{+}= \text{length}(words)$
 9:     end for
10: end for
11: for each word $w \in freq$ do
12:     Calculate word score: $\text{score}[w] = \frac{deg[w]}{freq[w]}$
13: end for
14: Initialize phrase score table $phrase\_scores$
15: for each candidate phrase $phrase \in P$ do
16:     Get the set of words in the phrase, denoted as $words$
17:     Calculate phrase score: $\text{phrase\_scores}[phrase] = \sum_{\forall w \in words} \text{score}[w]$
18: end for
19: Sort the candidate phrases in descending order based on $phrase\_scores$
20: Select the top-scoring phrases to form the keyword set $K_i$
```

RAKE (Rapid Automatic Keyword Extraction) is an unsupervised, domain-independent keyword extraction algorithm that identifies key phrases by analyzing the frequency of word appearances and their co-occurrences with other words in the text. We use the keywords extracted by RAKE to ensure that the student's generated summary retains the key information from the teacher's summary.

The core steps of the RAKE algorithm are shown in Algorithm 3. First, the text is split into candidate phrases based on stop words and punctuation. Next, the word frequency and *degree* (the number of candidate phrases in which the word appears) of each word in the candidate phrases are calculated. Then, a score is computed for each word, which is the ratio of its degree to its frequency. The score of a phrase is the sum of the scores of the words it contains. Finally, the candidate phrases are sorted according to their scores, and the highest-scoring phrases are selected as keywords.

Through the above steps, the RAKE algorithm can effectively extract key information from the teacher's summary S_i. These keywords are then used to compute the keyword matching loss L_{KW}, guiding the student model to generate more accurate summaries.

B Training Dataset Construction

B1 Knowledge Distillation Dataset Construction

We extract continuous text passages from Wikitext-103 to form a set of contexts $\{C_i\}_{i=1}^{N}$. Each context C_i consists of multiple consecutive sentences, ensuring sufficient content for meaningful summarization. We set the number of samples N to 2,400, which strikes a balance between dataset diversity and computational feasibility for training.

B2 QA Pair Generation:

To facilitate task-aware sentence-level pruning, we generate QA pairs based on the constructed dataset. The QA pairs help the model learn which sentences are most informative for answering specific questions, enhancing its ability to focus on essential content.The detailed procedure is presented in Algorithm 4.

Algorithm 4 QA Pair Generation Process

```
Input: Context $C_i$
Output: Question-Answer set $\{(Q_{ik}, A_{ik})\}_{k=1}^{M_i}$
 1: Select a context $C_i$ from the set $\{C_i\}$
 2: Split $C_i$ into individual sentences $\{s_{ij}\}_{j=1}^{N_i}$
 3: Initialize a downstream frozen large language model (LLM)
 4: Define a prompt to guide the LLM in generating questions and answers based on $C_i$
 5: for each context $C_i$ do
 6:     Generate question-answer pairs $\{(Q_{ik}, A_{ik})\}_{k=1}^{M_i}$ using the LLM
 7: end for
 8: Apply post-processing to filter the QA pairs:
        • Remove duplicate questions
        • Discard question-answer pairs where the answer is not in $C_i$
        • Filter out vague or irrelevant questions
 9: Compile the filtered QA pairs to form the dataset $\{(C_i, Q_{ik}, A_{ik})\}$
```

Summary Prompt in Training :
Generate a concise and informative summary of the following text. The summary should include the main points of the text while omitting unnecessary details. Ensure the summary maintains a clear logical structure, with well-organized paragraphs and highlighted key points. The summary should retain important nouns, key events, timelines, and relationships between individuals mentioned.
text:{...}

Question-Answer Generation Prompt in Trainging :
Generate questions closely related to the content of the following text and provide detailed answers. The questions should reflect the key points of the text and cover the main facts, data, and conclusions. Each question should be clear and concise, and the answers should be as specific as possible, based on the key information from the text.
text:{...}

Summary Prompt in Inference :
Generate a summary of the following text with a length proportionally reduced to $\tau_1$of the original text.
text:{...}

Figure 3: Instructions used in training and inference process.

For each context C_i, we generate M_i QA pairs. We set M_i proportionally to the length of C_i, typically generating 4 to 8 QA pairs per context. This approach ensures adequate coverage of the content within each context. Subsequently, we ultimately select two question-answer pairs for each context C_i. The total number of QA pairs generated is approximately 4800. This volume provides ample data for the model to learn meaningful associations between sentences and questions.

C Instructions Usage in Training and Inference

In Figure 3, we present the carefully designed prompts used during training and inference.

D Parameter α Verification Experiment

To verify the rationality of the value of parameter α, we conducted ablation experiments on the LongBench dataset (Single-Doc Subset) and tested the performance of α when $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$. The experimental settings were the same as those in the main experiment, with other parameters fixed and only the value of α adjusted. The results are shown in Table 5 below.

Experimental results show that when $\alpha = 0.5$, the model accuracy reaches 42.2%, which is consistent with the best result of the main experiment. Pure semantic similarity ($\alpha = 0$) or pure generation probability ($\alpha = 1$) both lead to a significant decrease in performance. Moreover, the accuracies when $\alpha = 0.25$ and $\alpha = 0.75$ are also lower than that of the balanced strategy. This verifies the necessity of the hybrid scoring mechanism: the generation probability of the language model (α term) can capture fine - grained signals related to the task, while the semantic similarity ($1 - \alpha$ term) ensures the retention of general information. The balance between the two significantly improves the integrity and task adaptability of the compressed content. Finally, $\alpha = 0.5$ is determined as the optimal parameter, and its design directly supports the efficiency and robustness of DisComp in the task - aware pruning stage.

Table 5: Ablation Study of Parameter α

α value	Accuracy
0	41.3
0.25	41.7
0.5	**42.2**
0.75	41.6
1	41.4