TAGCOS: Task-agnostic Gradient Clustered Coreset Selection for Instruction Tuning Data

Anonymous ACL submission

Abstract

Instruction tuning has achieved unprecedented success in NLP, turning large language models into versatile chatbots. However, the increasing variety and volume of instruction datasets demand significant computational resources. To address this, it is essential to extract a small and highly informative subset (i.e., *Coreset*) that achieves comparable performance to the full dataset. Achieving this goal poses nontrivial challenges: 1) data selection requires accurate data representations that reflect the training samples' quality, 2) considering the diverse nature of instruction datasets, and 3) ensuring the efficiency of the coreset selection algorithm for large models. To address these challenges, we propose ***T**ask-**A**gnostic **G**radient **C**lustered **CO**reset **S**election* (**TAGCOS**). Specifically, we leverage sample gradients as the data representations, perform clustering to group similar data, and apply an efficient greedy algorithm for coreset selection. Experimental results show that our algorithm, selecting only 5% of the data, surpasses other unsupervised methods and achieves performance close to that of the full dataset.

1 Introduction

Instruction tuning (Wei et al., 2022a; Ouyang et al., 2022) is the most important strategy for customizing Large Language Models (LLMs) for downstream tasks, which allows them to precisely understand human intentions and accurately generate responses in natural languages. Recently, many existing works (Wang et al., 2023a) expand the amount and diversity of instructions for instruction tuning to further enhance the LLM's capability. However, the increased quantity of the dataset also leads to significantly higher computational costs for instruction tuning. Meanwhile, Zhou et al. (2023) revealed that only 1,000 high-quality, human-created data samples could substantially improve the ability of LLMs to follow instructions, which suggests that there exists severe redundancy in current instruction datasets, and only a high-quality subset may suffice for achieving promising performance.

Hyper Parameter Items	Full (100%) Data	Selected (5%) Data
4 Learning Rate Options: 2e-5, 1e-4, 1e-5, 2e-4	322h	15.2h
4 Batch Size Options: 16, 32, 64, 128	315.3h	14.9h
4 Training Epochs Options: 1, 2, 3, 4	193.7h	9.2h
4 Max Lengths Options: 512, 768, 1024, 2048	248h	12.1h
4 Templates Options: User, ###Human, Question, [INST]	322h	15.2h
Selection Cost	-	231.1h
Total	**1401h**	**297.7h** (-1103.3h)
Evaluation Performance	49.58	48.35

Table 1: We provide a time cost comparison for the hyper parameter optimization experiments we actually conducted (using 4 A100-80G GPUs) on the full dataset (100%) and the selected subset (5%). The one-time effort of data selection can be utilized for multiple training runs across various requirements.

To address the above issue, selecting a small, highly informative subset (i.e., coreset) of training samples from the original dataset is a promising solution. As shown in table 1, this approach ensures that training on the coreset achieves performance comparable to the full dataset while significantly reducing iteration costs. However, coreset selection is challenging as it must not only consider the quality of individual samples, but also their importance within the entire subset. For example, if two high-quality samples are very similar, selecting only one may be sufficient. This global perspective on sample importance is crucial for the quality of the selected subset.

Current methods for coreset selection can be categorized into two main types: 1) Heuristic-based approaches (Marion et al., 2023; Li et al., 2023; Chen et al., 2023b; Lu et al., 2023), and 2) Optimization-based approaches (Borsos et al.,

2020; Zhou et al., 2022; Gao et al., 2023; Zhou et al., 2022). Heuristic-based methods use various heuristic scores to measure sample quality. For example, some assess data sample quality by ranking their corresponding perplexity score (Marion et al., 2023), while others score each sample using a powerful LLM (Chen et al., 2023b). These methods often rely on arbitrary heuristics that may not accurately evaluate sample quality and lack a comprehensive view of sample importance within the entire dataset, resulting in suboptimal performance. Optimization-based methods, on the other hand, typically frame the task as a bi-level optimization problem, requiring repeated optimization of both inner and outer loops. This approach incurs prohibitive costs, especially in the context of large language models (LLMs) that contain billions of parameters. Therefore, a coreset selection method that is applicable for LLMs is yet to be proposed.

In this paper, to address the above issues, we propose ***T**ask-**A**gnostic **G**radient **C**lustered **CO**reset **S**election* (**TAGCOS**), a coreset selection framework designed for LLM that is agnostic of its downstream tasks. Firstly, we use LLM's gradients as representation for each sample. Compared with representations based on model outputs, gradients effectively captures the information of how each sample affects the optimization direction of the LLM, which is the root cause of the model's final performance. Secondly, to perform coreset selection under a global view of the entire dataset, we show that coreset selection can be naturally formulated into a Submodular Function Maximization (SFM) problem. Then, noting that SFM is NP-hard (Bach et al., 2013) and naive solvers would be impractical when the dataset size is large, potentially leads to inferior solutions. This urges the development of efficient approximate optimizer, which is one of the main contributions of this work.

To be precise, we perform clustering on the gradient features over the dataset to decompose the SFM problem into several small-scaled subproblems to reduce the optimization difficulty. Lastly, we approximately solve each SFM subproblems via an efficient greedy approach named optimal matching pursuit (OMP) algorithm to perform coreset selection independently in each cluster in a fine-grained manner. This ensures a comprehensive coverage of the selected subset. Our theoretical analysis demonstrates that compared with the methods without our gradient clustering strategy, our method can achieve the comparable accuracy with a significantly smaller sized coreset.

In our experiment, we assessed the effectiveness of our method by selecting data from a combination of 17 popular instruction datasets (Wang et al., 2023a; Ivison et al., 2023), with a total of approximately 1 million data examples. By unsupervisedly selecting 5% of the original datasets, we obtained great performance on a range of evaluation benchmarks. Additionally, we confirmed the generalization of our method by applying the selected subset to various models.

Our main contributions are as follows:

- We verified that gradient features can serve as a good data representation that captures the essential information to measure the quality of instruction data.
- We propose Task-Agnostic Gradient Clustered Coreset Selection (TAGCOS), a coreset selection framework designed for LLM that is agnostic of its downstream tasks.
- Our experiment was conducted in a realistic setting, featuring 18 popular instruction datasets that include 1 million varied instruction data points. The practical results convincingly demonstrate the effectiveness of the entire pipeline.

2 Related Work

Instruction Tuning Data. Instruction tuning (Ouyang et al., 2022) has achieved unprecedented success in NLP, turning large language models into versatile chatbots (Chiang et al., 2023; Taori et al., 2023). Successful instruction tuning requires a powerful pre-trained base model as well as high-quality instruction datasets. For the powerful pre-trained base model, one usually selects a pre-trained LLM with more data and having more parameters, like Mistral (Jiang et al., 2023), Llama family models (Touvron et al., 2023). For high-quality instruction datasets part, it is expected that high-quality datasets are diverse and representative enough to adapt the LLM to potential downstream usage. With the development of instruction tuning, there are more and more instruction datasets. Usually, these datasets are either annotated by human or proprietary LLMs. Currently, instruction data generally contains these types: (1) datasets are created by researchers from existing NLP dataset and incorporate an instruction for existing input-output pairs, like Flan (Longpre et al., 2023; Wei et al.,

2022a), SuperNI (Wang et al., 2022), CoT (Wei et al., 2022b) and Orca (Mukherjee et al., 2023). (2) open-end text generation, e.g., multi-turn dialogue and instruction following. Several open-end text generation datasets are created by human, like Dolly (Databricks, 2023) and Oasst1 (Köpf et al., 2023). Others are generated by proprietary models or human interaction with these models, like Self-instruct (Wang et al., 2023b), Alpaca (Taori et al., 2023), Sharegpt (Chiang et al., 2023), Baize (Xu et al., 2023), GPT4-Alpaca (Peng et al., 2023) and Unnatural Instructions (Honovich et al., 2023). (3) instructions build for domain-specific skills, like Code-Alpaca (Chaudhary, 2023) for code completion. Given such a diverse collection of instruction dataset, the challenge for instruction tuning lies in ensuring the quality of these instructional data samples. Zhou et al. (2023) revealed that only several high-quality data samples could substantially improve the instruction tuning results. Thus, in this work, we aim to explore an automatic and unsupervised data selection technique to obtain the coreset for these instruction datasets.

LLM Data Selection. Since training LLM still request a lot of resources, data selection is often used for implementing efficient training. Also, several works (Zhou et al., 2023; Gunasekar et al., 2023) stress the importance of high-quality data and thus triggered more research works focus on data selection. One popular way to select data samples this is to use an extra LLM to evaluate data samples. Chen et al. (2023b); Lu et al. (2023) calls ChatGPT API to tag or evaluate the quality of the instruction data. Also, several works (Du et al., 2023; Bukharin and Zhao, 2023; Dong et al., 2023) make use of a reward model to assess the data quality. Wettig et al. (2024); Liu et al. (2024) intends to distill the preference of proprietary LLMs to small models for implementing efficient scalable data selection. This line of data selection methods is very expensive and suffers from interpretability. Another line of works focuses on using signals from the model itself to facilitate data evaluation and selection. Marion et al. (2023); Li et al. (2024) make use of perplexity or its variants to determine if a data sample is good or not. Xia et al. (2024); Pan et al. (2024) use the gradients and influence function to find the data sample that best matches the validation set for downstream tasks evaluation. Li et al. (2023); Cao et al. (2023) develops their own evaluation metric for assessing data samples. Compared to existing data selection approaches, our work focuses on selecting influential instruction data in a task-agnostic manner by utilizing LLM gradients on the dataset itself. Our method does not require a validation set and can be applied more broadly for data selection.

Algorithm 1 Coreset Selection

Require: A pretrained LLM θ, instruction tuning dataset $D = \{z_i \mid z_i = (s_i, c_i)\}_{i=1}^{N}$, target subset size M, training loss ℓ, gradient matching error function Err($\cdot$).

$\theta \leftarrow$ *FineTune*(D, θ) *# Warm up fine-tune with LoRA*
$\mathcal{G} \leftarrow \emptyset$
for each $z_i \in D$ **do**
  $g_i \leftarrow \nabla_\theta \ell(z; \theta)$ *# Calculate Sample Gradient*
  $\mathcal{G} \leftarrow \mathcal{G} \cup \{g_i\}$
end for
$\{\mathcal{C}_k\}_{k=1}^{K}, \{\mu_k\}_{k=1}^{K} \leftarrow$ K-means$(\mathcal{G}, K)$ *# Derive clusters and their centroids with K-means*
CoreSet $\leftarrow \emptyset$
for each cluster $\mathcal{C}_k$ with centroid μ_k **do**
  $r_k \leftarrow \frac{|\mathcal{C}_k|}{|\mathcal{D}|} \times M$ *# Derive subset size in k^{th} cluster*
  $\mathcal{C}_k^{sub}, w \leftarrow$ *OMP*$(\mathcal{C}_k, r_k, \text{Err}(\cdot))$ *# Derive the subset from k^{th} cluster*
  CoreSet $\leftarrow$ *CoreSet* $\cup \mathcal{C}_k^{sub}$
end for
Output: *CoreSet*

3 Method

To tackle the challenging coreset selection problem for LLM's instruction tuning dataset, we propose Task-Agnostic Gradient Clustered Coreset Selection (TAGCOS), a task-agnostic coreset selection approach that effectively and efficiently discovers the informative subset from a large instruction tuning dataset.

Notation. Assume we have a pretrained LLM θ and a giant and diverse instruction dataset $D := \{(s, c)_{(i)}\}_{i=1}^{N}$, where each data sample $z = (s, c)$ comprises an instruction s and a completion c. For each data sample, the loss $\ell(z; \theta)$ is defined as the cross entropy between the prediction distribution $p(\cdot \mid s)$ and the ground truth text response c. Since c often contains multiple tokens, $\ell(z; \theta)$ is calculated as the average of the token-wise cross entropy loss across the completion c. The notation θ_t refers to the model checkpoint at step t.

Problem Formulation. We first formulate the task into a gradient matching problem, i.e., the average gradient of the selected subset should approximate the gradient of the entire dataset. Intuitively, if the gradient is similar throughout all the training steps, the resulting model parameter should be closed to the model trained with the entire dataset.

Formally, given a giant and diverse dataset D, our goal is to select a subset $D_{sub} \subseteq D$ ($|D_{sub}| <$

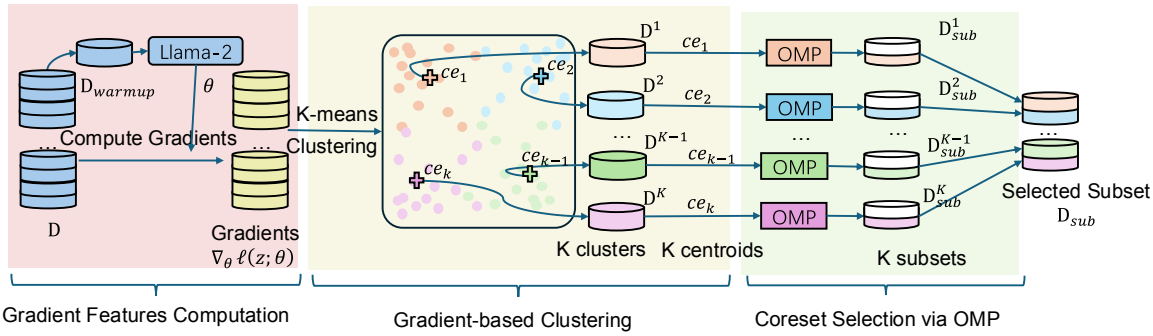

Figure 1: Illustration of the proposed **TAGCOS** pipeline. Our framework consists of three stages: 1) Gradient feature computation, which efficiently derive sample-wise gradients to use as data representation; 2) Gradient-based Clustering, which groups data with high similarity into the same groups; 3) Coreset selection via OMP, which efficiently selects the coresets from each cluster separately in a greedy manner.

$|D|$) containing the most informative training samples. We expect that the gradients produced by the full training dataset $\sum_{z\in D} \nabla_\theta \ell(z;\theta)$ can be replaced by the gradients produced by a subset $\sum_{z\in D_{sub}} \nabla_\theta \ell(z;\theta)$ with minimal difference:

$$\min_{\mathbf{w}, D_{\text{sub}}} \text{Err}\Big(\nabla_\theta \mathcal{L}(D;\theta), \frac{1}{\|\mathbf{w}\|_1} \sum_{z\in D_{\text{sub}}} w_z \nabla_\theta \ell(z;\theta)\Big)$$
$$\text{s.t.} \quad D_{\text{sub}} \subseteq D, \quad w_z \geq 0, \quad |D_{sub}| \leq M \tag{1}$$

where $\mathcal{L}(D;\theta) = \frac{1}{|D|}\sum_{z\in D} \nabla_\theta \ell(z;\theta)$, w is the subset weight vector, $\|\mathbf{w}\|_1$ is the sum of the absolute values and $\text{Err}(\cdot,\cdot)$ measures the distance between two gradients. Note that w could be either varibales, which leads to weighted training on the selected subset, or with discrete values, which reduces to regular training on the coreset.

Issues and Solution. However, due to the high diversity of the large-scale instruction tuning dataset, simply conducting selection over the entire dataset potentially causes over-sampling in certain domains and under-sampling in others. To address this, we introduce clustering to ensure balanced sampling. By splitting the dataset into clusters and selecting samples from each cluster, we ensure a more even distribution across different domains.

Overall, as illustrated in algorithm 1, the process for coreset construction could be summarized as follows: (1) **compute the gradient features** $\mathcal{G} = \{g_i \mid g_i = \nabla_\theta \ell(z;\theta)\}_{i=1}^N$. It is worth noting that the "graident" here actually refers to *the first-order approximation* of training dynamics, which is actually a little different from gradient itself. For briefness, we term this **the first-order approximation** as gradient here. More exactly, we compute the low-dimensional approximations of gradient features for each data samples z over the whole dataset D; (2) **perform gradient-based clustering**, we perform k-means clustering (Hartigan and Wong, 1979) given the gradients features and get k clusters and corresponding centroids ce for each cluster, which effectively gathers the samples with similar characteristics into one cluster; (3) **coreset selection via Optimal Matching Pursuit**, we compute the data samples matches best with the centroids in each cluster with an orthogonal matching pursuit algorithm (Killamsetty et al., 2021).

3.1 Gradient Features Computation

We perform an efficient gradient feature approximation computation over the entire dataset. To speed up gradient computation, we use LoRA (Hu et al., 2022) and random projections (Park et al., 2023) to reduce the dimensions in gradient features.

Adam Gradient Computation Function. The gradients based on Adam optimizer Kingma and Ba (2015) can be computed with these steps:

$$\theta^{t+1} - \theta^t = -\eta_t g_i(z, \theta^t) \tag{2}$$

$$g_i(z,\theta^t) \triangleq \frac{m^{t+1}}{\sqrt{v^{t+1}} + \epsilon} \tag{3}$$

$$m^{t+1} = \left(\beta_1 m^t + (1-\beta_1)\nabla \ell(z;\theta^t)\right)/(1-\beta_1^t) \tag{4}$$

$$v^{t+1} = \left(\beta_2 v^t + (1-\beta_2)(\nabla \ell(z;\theta^t))^2\right)/(1-\beta_2^t) \tag{5}$$

where β_1 and β_2 are hyperparameters, and ϵ is a small constant. $g_i(z,\theta^t)$ represents the first-order expansion for the Adam dynamics, requiring model gradients and optimizer states from the training process. Warmup training on a subset of the dataset

provides the necessary checkpoints for these computations. As mentioned above, we will sample checkpoints before convergence to provide a more accurate gradient estimation.

Warmup Training and Projection. LoRA (Hu et al., 2022) is used to reduce the number of trainable parameters and accelerate the inner products in $g_i(z, \theta^t)$. LoRA freezes the pre-trained weights and adds a low-rank adaptor to the selected fully connected layers. We use LoRA to perform instruction tuning on pre-trained base model (e.g., LLAMA-2-7B) on a random subset $\mathcal{D}_{\text{warmup}} \subseteq \mathcal{D}$ for N epochs, checkpointing the model after each epoch to store $\{\theta_i\}_{i=1}^N$. The gradient when training with LoRA, denoted $\widehat{\nabla}\ell(\cdot;\theta) \in \mathbb{R}^P$, is much lower dimensional than the model itself; for example, in LLAMA-2-7B, $\widehat{\nabla}\ell(\cdot;\theta)$ is less than 2% the size of θ. We use $\widehat{\nabla}\ell(\cdot;\theta)$ to compute the Adam update and denote it as $\widehat{g}_i(\cdot,\theta)$.

In order to further accelerate the computation, a random projection, Rademacher distribution (Johnson, 1984) (i.e., $\Pi_{ij} \sim \mathcal{U}(\{-1,1\})$), is introduced to further reducing the feature dimension. In total, we compute gradient features for each data sample z with $\widetilde{g}_i(z,\cdot) = \Pi^\top \widehat{g}_i(z,\cdot)$.

3.2 Gradient-based Clustering

Due to the diversity of instruction tuning dataset, direct sampling over the entire dataset may not cover all the regions, since the training samples from each domain are not evenly distributed. To further improve the effectiveness and robustness of data selection, we divide the entire dataset into several clusters and then perform gradient matching algorithm on each cluster itself. With the gradient features g_i from the above step, we conduct K-means clustering on them to assign each data sample into a cluster $\{\mathcal{C}_k\}_{k=1}^K$. Also, we can obtain cluster centroids $\{\mu_k\}_{k=1}^K$ of these clusters during the clustering process, where each centroid shares the dimension with gradient features.

3.3 Coreset Selection via Optimal Matching Pursuit

In each cluster, we hope to get the subset that minimizes the difference between the selected subset and the whole cluster. Instead of doing heuristic selection like selecting all the instances with shortest distance with cluster centroids, we formalize this as an optimization problem and introduce an orthogonal matching pursuit (OMP) algorithm (Killamsetty et al., 2021; Elenberg et al., 2016) to solve it. Similar with equation 1, our objective is to minimize the difference between selected D_{sub}^k in k-th cluster and the whole cluster D^k,

$$\text{Err}(\mathbf{w}^k, D_{sub}^k; D^k) \triangleq \left\| \sum_{z \in D_{\text{sub}}^k} w_z^k \widehat{\nabla}_\theta \ell(z;\theta) - \frac{1}{|D^k|} \sum_{z \in D^k} \widehat{\nabla}_\theta \ell(z;\theta) \right\| \tag{6}$$

Considering the regularization coefficient λ, we can have $\text{Err}_\lambda(\mathbf{w}, D_{sub}^k; D^k)$ as:

$$\text{Err}_\lambda(\mathbf{w}, D_{sub}^k; D^k) \triangleq \text{Err}(\mathbf{w}, D_{sub}^k, D^k) + \lambda \|\mathrm{w}\|^2 .$$

Here, we approximately regard the centroids of each cluster as the average gradients of the whole cluster,

$$\frac{1}{|D^k|} \sum_{z \in D^k} \nabla_\theta \ell(z;\theta) = ce_k. \tag{7}$$

We next study the optimization algorithm for solving equation 6. Our goal is to minimize $\text{Err}_\lambda(\mathbf{w}, D_{sub}^k; D^k)$ subject to the constraint $D_{sub}^k : |D_{sub}^k| < d_k$. We can convert this into maximization problem over the set D_{sub}^k , i.e.,

$$\max_{D_{sub}^k} F_\lambda(D_{sub}^k; D^k), \tag{P-k}$$
$$s.t., |D_{sub}^k| \le d_k \text{ and } D_{sub}^k \subseteq D^k,$$

Here the objective $F_\lambda(D_{sub}^k; D^k)$ is defined as

$$F_\lambda(D_{sub}^k; D^k) \triangleq L_{\max}^k - \min_{\mathbf{w}} \text{Err}_\lambda(\mathbf{w}, D_{sub}^k; D^k),$$

where $L_{\max}^k$ is a constant to make the objective non-negative. Note that we minimize $\text{Err}_\lambda(D_{sub}^k)$ subject to the constraint $D_{sub}^k : |D_{sub}^k| < d_k$ until $\text{Err}_\lambda(D_{sub}^k) < \epsilon$, where ϵ is the tolerance level and tk is the target num of samples in the selected subset. Note that minimizing $\text{Err}_\lambda(D_{sub}^k)$ is equivalent to maximizing $F_\lambda(D_{sub}^k)$. Given this, we use OMP to solve this optimization problem, details of OMP are presented in Algorithm 2.

In each cluster k, we select data samples that can minimize $\text{Err}_\lambda(D_{sub}^k)$ with the above-described OMP algorithm. After finishing the selection on each cluster, we combine the selected subset D_{sub}^k to be D_{sub} and use it to train the target model.

Algorithm 2 OMP

Require: full dataset D, Target subset size M, error function Err($\cdot$)
$D_{sub} \leftarrow \emptyset$
$r \leftarrow \nabla_{\mathbf{w}} \text{Err}_\lambda(\mathbf{w}, D_{sub}; D)\big|_{\mathbf{w}=0}$
while $|D_{sub}| \leq M$ **and** $\text{Err}_\lambda(\mathbf{w}, D_{sub}; D) \geq \epsilon$ **do**
  $e = \arg\max_j |r_j|$
  $D_{sub} \leftarrow D_{sub} \cup \{e\}$
  $\mathbf{w} \leftarrow \arg\min_{\mathbf{w}} \text{Err}_\lambda(\mathbf{w}, D_{sub}; D)$
  $r \leftarrow \nabla_{\mathbf{w}} \text{Err}_\lambda(\mathbf{w}, D_{sub}; D)$
end while
return $D_{sub}, \mathbf{w}$

4 Theoretical Analysis

In this section, we analyse the benefits of our gradient clustering in coreset selection. The general conclusion is that coreset selection problem formulated in Problem (P) is essentially a Submodular Function Maximization (SFM) problem (Bach et al., 2013), which is NP-hard. Solving large-scaled SFM problems is extremely challenging, potentially leads to inferior solution. For the discussion about the assumption here, one may refer to appendix B. Our gradient clustering strategy naturally decomposes the original problem into several small scaled problems, significantly reduces the difficulty in optimization, making finding solutions with high-precision possible. The detailed results are presented in the following theorems. These theorems are adapted from the classical analysis on OMP, which can be found in the studies (Elenberg et al., 2018; Wolsey, 1982). We adopt them to understand the superiority of our coreset selection approach.

To unify the problems of coreset selection with and without clustering, we extend the problem (P-k) as follows:

$$\max_{D_{sub}} F_\lambda(D_{sub}; D), \quad \text{(P)}$$
$$s.t., |D_{sub}| \leq M \text{ and } D_{sub} \subseteq D,$$

where D_{sub} and D are the coreset and the full dataset, respectively. c is the constant to control the coreset size.

Lemma 1 *If the coreset size $|D_{sub}| \leq c$ and $\max_{z \in D} \|\nabla_\theta \ell(z;\theta)\|_2 \leq G$, then $F_\lambda(D_{sub}; D)$ is γ_D-weakly submodular with $\gamma_D = \frac{\lambda}{\lambda + MG^2}$.*

Theorem 1 *If $\max_{z \in D} \|\nabla_\theta \ell(z;\theta)\|_2 \leq G$ and $\max_{z \in D^k} \|\nabla_\theta \ell(z;\theta)\|_2 \leq G_k$ for cluster k. Let D^*_{sub} and D^{k*}_{sub} be the optima of Problems P and P − k, with $k = 1, \ldots, K$. Then, the followings hold:*

(i) *For problem (P), OPM runs with stopping criteria $F_\lambda(D_{sub}; D) \leq \epsilon$ achieves set D_{sub} with $|D_{sub}| \leq \frac{|D^*_{sub}|}{\gamma_D} \log(\frac{L_{max}}{\epsilon})$.*

(ii) *For problem (P-k), OPM runs with stopping critia $F_\lambda(D^k_{sub}; D^k) \leq \epsilon_k$ achieves set D^k_{sub} with $|D^k_{sub}| \leq \frac{|D^{k*}_{sub}|}{\gamma_{D^k}} \log(\frac{L^k_{max}}{\epsilon_k})$.*

Since $\gamma_D = \frac{\lambda}{\lambda + MG^2}$ and $\gamma_{D^k} = \frac{\lambda}{\lambda + d_k G_k^2}$ with $M = \sum_{k=1}^K d_k$, it can be expected that $\gamma_D \ll \gamma_{D^k}$. Noting that a proper clustering method would make $D^*_{sub} \approx \cup_{k=1}^K D^{k*}_{sub}$ and it is reasnonable to set $\frac{L^k_{max}}{\epsilon_k} \approx \frac{L_{max}}{\epsilon}$ to ensure comparable precisions. Thus the above theorem demonstrates that

$$\sum_{k=1}^K \frac{|D^{k*}_{sub}|}{\gamma_{D^k}} \log(\frac{L^k_{max}}{\epsilon_k}) \ll \frac{|D^*_{sub}|}{\gamma_D} \log(\frac{L_{max}}{\epsilon}).$$

That is, to achieve comparable accuracy, the union of the coreset selected from each cluster can be much smaller than that from the whole datasets, which verifies the benefits of gradient clustering. This is also consistent with our experimental observation. i.e., the running time of OMP without gradient clustering is significantly longer than that with gradient clustering.

5 Experiment

In this section, we conduct experiments to answer the following research questions:

- Does TAGCOS achieve superior performance over other unsupervised selection methods? (Table 2, Table 3)
- How effective is the generalization of TAGCOS, and can it be transferred to different models? (Table 4)
- What is the best configuration for TAGCOS, including the selection proportion, the number of clusters, and the selection of gradient checkpoints? (Table 5, Table 6, Table 7)

5.1 Setup

To illustrate that TAGCOS is task agnostic, we chose diverse tasks for both training and evaluation. For the training set, we combined 17 popular instruction datasets totaling 1,068,549 examples,

	TydiQA	MMLU	BBH	average
Uniform	52.08	46.9	41.39	46.79
Hardest	51.58	45.68	38.15	45.13
Perplexity	51.66	46.89	40.74	46.43
K-Center	38.83	48.73	41.48	43.01
OMP	53.64	46.10	40.47	46.82
TAGCOS	**52.78** +0.7	**48.01** +1.11	**44.26** +2.87	**48.35** +1.65

Table 2: Evaluations used TydiQA, MMLU, and BBH benchmarks. Results reflect Llama-2 7B models trained on 5% of data selected by each method. Note that we tested K-Center and OMP data selection methods with gradient embedding on 17 instruction datasets.

following Wang et al. (2023a); Ivison et al. (2023). These datasets vary in format and reasoning tasks, with annotations by humans or the OpenAI API. For details, please refer to Appendix G.

For evaluation, we selected **TydiQA** (Clark et al., 2020), **MMLU** (Hendrycks et al., 2020), and **BBH** (Suzgun et al., 2022). For the Implementation Details, please refer to appendix A.

5.2 Experimental Results

Baseline. The main experiment results are presented in Table 2. Several baselines were considered for comparison: (1) **Uniform**: randomly selecting the data samples from the original dataset. (2) **Hardest Sampling**: select the data samples with the highest perplexity. (3) **Perplexity Sampling** (Marion et al., 2023; Marcus et al., 1993): select the data samples with the lowest perplexity. (4) **K-Center-Greedy with different representations** (Chen et al., 2023a): converting instruction data into gradient embedding vectors, performing K-means clustering, and selecting samples by iteratively choosing the one closest to the cluster center among the remaining instances. For details about other clustering methods, we include them in the appendix 11. (5) **OMP** (Killamsetty et al., 2021): using the OMP algorithm over the entire dataset, with the mean gradient feature across the dataset as the matching target.

Main Experiments. As shown in Table 2, TAGCOS achieves the best performance across all tasks, confirming its effectiveness in data selection for instruction tuning. TAGCOS is the only baseline that consistently performs well. Although K-Center$_{Grad}$ excels on the MMLU benchmark, it fails on TydiQA and is equivalent to uniform sampling on BBH, underscoring TAGCOS's robustness.

Effectiveness of each Component in TAGCOS. The key difference between TAGCOS and Grad+K-Center lies in their selection mechanisms. While K-means clustering on gradient features can achieve strong results on individual benchmarks, it is insufficient for consistent overall performance. This further demonstrates the effectiveness of the OMP coreset selection algorithm. Compared to OMP, which does not use clustering, TAGCOS delivers better results. This reinforces our perspective that clustering is essential for managing the diversity in instruction datasets.

	TydiQA	MMLU	BBH	average
Uniform	52.08	46.9	41.39	46.79
K-Center				
BERT	50.05	47.16	39.91	45.7
Llama	52.72	46.07	39.07	45.95
Grad	38.83	48.73	41.48	43.01
K-Center+OMP				
BERT	52.21	45.65	42.95	46.93
Llama	52.01	46.04	44.26	46.08
Grad	**52.78**	**48.01**	**44.26**	**48.35**

Table 3: We compare the selection results of clustering and optimal matching pursuit methods across various embedding spaces. All outcomes are based on 5% of the data samples chosen by the respective methods and trained using Llama-2 7B models.

Gradient Features vs. Other Embeddings. As shown in table 3, we evaluated the K-Center algorithm with various data representation schemes, including BERT, Llama, and Gradient. In the absence of a selection mechanism, gradient features exhibits great improvement over MMLU and BBH while dropping a lot on TydiQA. We attribute the multilingual features to TydiQA which evaluates some low-resource languages performance. As shown in appendix C, we can learn that K-center with gradient overfits to English. We argue that the reason is OMP considers the entire cluster distribution to find a representative subset, whereas

K-center focuses on sample-centroid distances.

5.3 Ablation Study and Analysis

Performance of TAGCOS on Different Models. Table 4 demonstrates that the dataset generated by the Llama-2-7B model can be effectively utilized to train a superior Mistral-7B instruction model. By leveraging the datasets selected by TAGCOS on the Llama-2-7B model, the trained Mistral-7B model shows significant improvements over uniform selection methods, consistently outperforming its counterparts. This highlights TAGCOS's ability to identify transferrable and valuable data samples, indicating its potential for future proxy data selection tasks.

	TydiQA	MMLU	BBH	Average
Llama-2 7B				
Uniform	52.08	46.9	41.39	46.79
TAGCOS	52.78	48.01	44.26	48.35
Mistral 7B				
Uniform	57.59	61.34	56.48	58.47
TAGCOS	61.49	61.79	57.87	60.38

Table 4: Experiments showing the impact of transferring TAGCOS-selected datasets from Llama-2 7B to Mistral-7B. Consistent improvement on TydiQA, MMLU, and BBH benchmarks demonstrate the transferability.

5% data can achieve comparable results with full dataset. Table 5 reveals that training with only 5% of the data selected by TAGCOS results in performance comparable to that of the entire dataset. This can be attributed to the presence of noisy samples in the full dataset, which are less effective for fine-tuning.

prop	TydiQA	MMLU	BBH	Average
5%	52.78	48.01	44.26	48.35
20%	51.76	48.87	42.95	47.86
25%	52.13	49.95	43.33	48.47
100%	51.44	52.96	44.35	49.58

Table 5: Results of experiments with different selection proportions using the Llama-2 7B model.

How to determine the cluster numbers. Table 6 shows that the ideal cluster number for our setup is 100. Fewer clusters, especially less than the original dataset size of 18, fail to achieve good results. Additionally, merely increasing the number of clusters does not ensure improved performance. TAGCOS tends to degrade to plain OMP as the number of clusters increases. When the cluster count matches the number of samples, the performance is identical to plain OMP. For more detailed results for cluster numbers, please refer to 12.

# Cluster	TydiQA	MMLU	BBH	Average
10	54.04	47.71	40.00	47.25
20	52.58	45.76	41.11	46.48
50	54.84	47.09	42.96	48.30
100	52.78	48.01	44.26	48.35
200	52.57	46.87	42.87	47.44

Table 6: Experimental results show the results on selecting different numbers of clusters.

Selecting early stopped checkpoints for computing gradients. In Table 7, "Checkpoints with significant loss reduction" means all the warmup checkpoint used for computing gradient features comes from the steps before convergence. "Evenly spaced checkpoints" represents that these checkpoints are sampled across the entire training process evenly. We argue that "early-selecting", i.e., sample checkpoints from steps before convergence, works better since the gradients before convergence provide more effective reactions for data samples for training. The results in this table also support this idea. In total, it is better to have a warmup checkpoint sampled from steps before convergence to get better results on TAGCOS.

Selected Steps	TydiQA	MMLU	BBH	Average
Checkpoints with significant loss reduction				
1,10,20,30	52.78	48.01	44.26	48.35
1,5,10,20	53.36	47.74	42.15	47.75
1,5,10,15	53.29	47.08	43.26	47.84
Evenly spaced checkpoints				
1,8348, 16696, 25044	53.14	47.16	39.54	46.61

Table 7: In the study of warmup checkpoint selection, minimal loss reduction occurs after step 40, out of a total of 33,392 steps.

6 Conclusion

This paper focuses on the effective selection of coresets for LLMs in instruction tuning. We utilize gradient features to act as data representations, which indicate the influence of each data sample on the training process. Additionally, to handle diverse collections of instruction data and ensure selection efficiency, we propose clustering similar data and applying an efficient greedy algorithm for selection. Our experimental results demonstrate the effectiveness of the entire pipeline.

7 Limitation

Despite its impressive performance, TAGCOS is bottlenecked by gradient feature estimation. The gradient feature computation stage limits its scalability to larger datasets. To effectively run TAGCOS on extensive datasets, improvements in the efficiency of gradient computation are needed.

References

Francis Bach et al. 2013. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends® in Machine Learning*, 6(2-3):145–373.

Zalán Borsos, Mojmir Mutny, and Andreas Krause. 2020. Coresets via bilevel optimization for continual learning and streaming. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Alexander Bukharin and Tuo Zhao. 2023. Data diversity matters for robust instruction tuning. *CoRR*, abs/2311.14736.

Yihan Cao, Yanbin Kang, and Lichao Sun. 2023. Instruction mining: High-quality instruction data selection for large language models. *arXiv preprint arXiv:2307.06290*.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. GitHub repository.

Hao Chen, Yiming Zhang, Qi Zhang, Hantao Yang, Xiaomeng Hu, Xuetao Ma, Yifan Yanggong, and Junbo Zhao. 2023a. Maybe only 0.5% data is needed: A preliminary exploration of low training data instruction tuning. *CoRR*, abs/2305.09246.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. 2023b. Alpagasus: Training A better alpaca with fewer data. *CoRR*, abs/2307.08701.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. Blog post.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *TACL*.

Databricks. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm. Blog post.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. RAFT: reward ranked finetuning for generative foundation model alignment. *CoRR*, abs/2304.06767.

Qianlong Du, Chengqing Zong, and Jiajun Zhang. 2023. Mods: Model-oriented data selection for instruction tuning. *CoRR*, abs/2311.15653.

Ethan R Elenberg, Rajiv Khanna, Alexandros G Dimakis, and Sahand Negahban. 2018. Restricted strong convexity implies weak submodularity. *The Annals of Statistics*, 46(6B):3539–3568.

Ethan R. Elenberg, Rajiv Khanna, Alexandros G. Dimakis, and Sahand N. Negahban. 2016. Restricted strong convexity implies weak submodularity. *CoRR*, abs/1612.00804.

Jiahui Gao, Renjie Pi, Yong Lin, Hang Xu, Jiacheng Ye, Zhiyong Wu, Weizhong Zhang, Xiaodan Liang, Zhenguo Li, and Lingpeng Kong. 2023. Self-guided noise-free data generation for efficient zero-shot learning. *Preprint*, arXiv:2205.12679.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. 2023. Textbooks are all you need. *CoRR*, abs/2306.11644.

John A Hartigan and Manchek A Wong. 1979. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. In *International Conference on Learning Representations* (ICLR).

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2023. Unnatural instructions: Tuning language models with (almost) no human labor. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 14409–14428. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew E. Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy,

and Hannaneh Hajishirzi. 2023. Camels in a changing climate: Enhancing LM adaptation with tulu 2. *CoRR*, abs/2311.10702.

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

William B Johnson. 1984. Extensions of lipshitz mapping into hilbert space. In *Conference modern analysis and probability, 1984*, pages 189–206.

KrishnaTeja Killamsetty, Durga Sivasubramanian, Ganesh Ramakrishnan, Abir De, and Rishabh K. Iyer. 2021. GRAD-MATCH: gradient matching based data subset selection for efficient deep model training. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 5464–5474. PMLR.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. 2023. Openassistant conversations - democratizing large language model alignment. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. 2024. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. *CoRR*, abs/2402.00530.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2023. From quantity to quality: Boosting LLM performance with self-guided data selection for instruction tuning. *CoRR*, abs/2308.12032.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2024. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 22631–22648. PMLR.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. #instag: Instruction tagging for analyzing supervised fine-tuning of large language models. *CoRR*, abs/2308.07074.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguistics*, 19(2):313–330.

Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. 2023. When less is more: Investigating data pruning for pretraining llms at scale. *CoRR*, abs/2309.04564.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of GPT-4. *CoRR*, abs/2306.02707.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

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

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. 2023. TRAK: attributing model behavior at scale. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 27074–27113. PMLR.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with GPT-4. *CoRR*, abs/2304.03277.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

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. GitHub repository.

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

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023a. How far can camels go? exploring the state of instruction tuning on open resources. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13484–13508. Association for Computational Linguistics.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Gary Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 5085–5109. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022a. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. 2024. Qurating: Selecting high-quality data for training language models. *CoRR*, abs/2402.09739.

Laurence A Wolsey. 1982. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2(4):385–393.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. 2024. LESS: selecting influential data for targeted instruction tuning. *CoRR*, abs/2402.04333.

Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2023. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. *arXiv preprint arXiv:2304.01196*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.

Xiao Zhou, Renjie Pi, Weizhong Zhang, Yong Lin, Zonghao Chen, and Tong Zhang. 2022. Probabilistic bilevel coreset selection. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 27287–27302. PMLR.

A Implementation Details

In this work, we performed warmup training on a randomly selected 5% of the dataset for 4 epochs and computed 8192-dimensional gradient features on the full dataset D. The learning rate for warmup training was set to 2e-5, with a batch size of 32. Using these gradient features, we selected 5% of the original dataset using our selection methods, totaling approximately 53,427 samples. We used 100 clusters for K-means clustering and set the

OMP algorithm tolerance at 0.01. After obtaining the subset, we fine-tuned the Llama-2-7B (Touvron et al., 2023) and Mistral-7B (Jiang et al., 2023) models using LoRA (Hu et al., 2022) to reduce memory usage. For LoRA training, we used the AdamW optimizer with a learning rate of 2e-5 and 4 epochs. The context length was set to 1,024, with a batch size of 32.

B Assumpation about the Selected Optimal Subset

Actually, our result can hold with a weaker assumption. To be precise, from the definition of γ_D and γ_{D^k}, when the problems $P - k$ are not too unbalanced, it can be expected that $\gamma_D \leq \gamma_{D^k}/\kappa_\gamma$ with a large number $\kappa_\gamma > 0$. As problem P is decomposed into the subproblems by clustering, it is natural to assume that the optimal solution D_{sub}^{k*} do not have significant overlap, that is $\sum_{k=1}^{K} |D_{sub}^{k*}| \leq \kappa_D |D_{sub}^{*}|$ with $1 \leq \kappa_D \ll \kappa_\gamma$. Thus, letting $\frac{L_{max}^k}{\epsilon_k} \approx \frac{L_{max}}{\epsilon}$, we have

$$\begin{aligned}
&\sum_{k=1}^{K} \frac{|D_{sub}^{k*}|}{\gamma_{D^k}} \log(\frac{L_{max}^k}{\epsilon_k}) \\
\leq &\frac{1}{\kappa_\gamma \gamma_D} \log(\frac{L_{max}}{\epsilon}) \sum_{k=1}^{K} |D_{sub}^{k*}| \\
\leq &\frac{\kappa_D |D_{sub}^{*}|}{\kappa_\gamma \gamma_D} \log(\frac{L_{max}}{\epsilon}) \\
\ll &\frac{|D_{sub}^{*}|}{\gamma_D} \log(\frac{L_{max}}{\epsilon}).
\end{aligned}$$

Table 8 are some empirical results about quality and speed differences with and without clustering. The results show that clustering improves quality with minimal added time.

Method	TydiQA	MMLU	BBH	Average	Time
without Clustering	53.64	46.10	40.47	46.82	-
with Clustering	52.78	48.01	44.26	**48.35**	+1.2h

Table 8: Comparison of Methods with and without Clustering

C Language Overfitting of K-Center_grad

We argue that the reason that K-Center_grad significantly deteriorates only for TydiQA is that K-center_grad overfit to English. As shown in table 9, OMP considers the entire cluster distribution to find a representative subset, whereas K-center focuses on sample-centroid distances. Here is the statistic of recall in English and several languages for K-Center_BERT, K-Center_Llama and K-Center_Grad.

Language	BERT	Llama	Grad
Bengali	24.7	30.0	23.8
English	66.5	66.5	**71.5**
Korean	80.5	78.5	78.0

Table 9: Details of several language results for K-Center_BERT, K-Center_Llama and K-Center_Grad.

D Detailed Hidden Time Cost of TAGCOS

Task	Time
Warmup Training	80.5h
Gradient Feature Computation	144.8h
Gradient-based Clustering	1.2h
Coreset Selection via OMP	0.8h
Selected Subset Training	3.8h
Total	231.1h

Table 10: TAGCOS detailed time summary.

Table 10 shows that the gradient feature computation stage is the most time-consuming, highlighting the need for future optimization.

E Clustering Exploration

Other cluster methods. We tried several clustering methods different from K-means like DBSCAN, Agglomerative clustering, BIRCH and Ward hierarchical clustering here 11. We find that they can not achieve convergence and failed to give clustering results. We suggest that the reason is that these other clustering methods are not suitable for this extremely large dataset (with > 1,000,000 examples).

Method	TydiQA	MMLU	BBH	Average	Convergence
minibatch k-means	53.63	46.2	41.48	47.1	True
Agglomerative	-	-	-	-	False
BIRCH	-	-	-	-	False
Ward Hierarchical	-	-	-	-	False
DBSCAN	-	-	-	-	False

Table 11: Clustering Method Performance and Convergence

Number of Clusters. As shown in table 12, the cluster number lies in [50,100] can all provide great

results, which proves that the effectiveness of the proposed method is robust to choices of # of clusters.

#Cluster	TydiQA	MMLU	BBH	Average
50	54.84	47.09	42.96	48.30
60	54.56	47.46	42.22	48.08
70	53.19	47.78	43.56	48.18
80	54.21	47.00	43.06	48.09
90	53.05	47.45	44.01	48.17
100	52.78	48.01	44.26	48.35

Table 12: Performance Metrics Across Different Clusters

OMP selected data samples in each cluster. Table 13, Table 14 and Table 15, show the detailed distribution of how the selected examples fall into different clusters.

F OMP further exploration

OMP is different with different embedding spaces. As shown in table 16, it is worth noting that Llama+Cluster+OMP costs much more time on OMP, which reflects that Llama embedding is not very well on telling different data examples apart.

OMP with Clustering showing differences. According to our formulation, this coreset selection problem is a submodular function maximization problem. As referenced in [1], it is an NP-hard problem, typically requiring polynomial-time approximation algorithms like local search, which aligns with our approach here.

we have compiled statistics for the OMP selection results (OMP Selected Data) on the full dataset and its intersection with cluster results in table 17. To better illustrate how the selected examples are distributed across different clusters, we include a baseline with the same 53,427 samples, artificially evenly distributed among the 10 categories with the largest data volumes (reDistributed in Top 10 Categories):

- Entropy: The entropy value is higher for the OMP Selected Data , indicating a more evenly distribution.
- Gini Coefficient: The Gini coefficient is higher for the OMP Selected Data , suggesting slightly more inequality.
- Coefficient of Variation: The coefficient of variation is higher when data is distributed in the top 10 categories, showing greater relative differences among these categories.

G Training Dataset Details

In this section, we provide the detailed sources, statistics and licenses of each training dataset used in our experiment, which is shown in table 18. We conduct coreset selection from a mixture of 17 instruction tuning datasets with various scales and properties, which demonstrates superior effectiveness compared with baseline approaches.

Cluster ID	Cluster Size	Union with OMP	Union with Baseline redistributed in Top 10 Categories
0	90019	4978	5343
1	23935	1349	5343
2	14912	1499	5343
3	9731	498	5343
4	45696	4694	5343
5	5760	247	0
6	5620	241	0
7	6634	440	0
8	2578	114	0
9	10192	604	5343
10	8529	416	5343
11	1296	70	0
12	53419	2776	5343
13	1567	114	0
14	5148	143	0
15	30084	702	5343
16	11126	673	0
17	635	38	0
18	7953	332	0
19	1404	22	0
20	9763	504	0
21	10499	363	0
22	3017	105	0
23	7239	265	0
24	2623	96	0
25	1369	69	0
26	20374	764	0
27	18827	1114	0
28	988	17	0
29	1416	11	0
30	10861	468	0
31	12974	568	0
32	11466	131	0
33	4527	106	0
34	4809	215	0
35	59075	3290	5343
36	39980	2128	5343
37	1345	16	0
38	678	37	0
39	190	2	0
40	5770	307	0
41	3174	108	0
42	10373	894	0
43	2478	115	0
44	6649	243	0
45	299	5	0

Table 13: Cluster Information and Union Metrics

Cluster ID	Cluster Size	Union with OMP	Union with Baseline redistributed in Top 10 Categories
46	3562	203	0
47	12200	214	0
48	10348	630	0
49	9308	443	0
50	3751	136	0
51	1	0	0
52	11090	625	0
53	30081	1307	0
54	21439	191	0
55	19520	810	0
56	8250	382	0
57	6886	242	0
58	3512	167	0
59	3774	206	0
60	3570	219	0
61	1444	29	0
62	919	40	0
63	2	0	0
64	11750	637	0
65	7532	682	0
66	23828	1095	0
67	11880	669	0
68	6717	165	0
69	1999	103	0
70	4106	214	0
71	2753	128	0
72	1409	63	0
73	15517	524	0
74	14888	692	0
75	478	24	0
76	15419	681	0
77	530	51	0
78	1401	73	0
79	40549	2534	5343
80	2496	51	0
81	12687	393	0
82	1019	19	0
83	5147	70	0
84	2012	99	0
85	5495	257	0
86	4158	131	0
87	159	2	0
88	4845	243	0
89	9668	376	0
90	40217	2115	5343

Table 14: Cluster Information and Union Metrics

Cluster ID	Cluster Size	Union with OMP	Union with Baseline redistributed in Top 10 Categories
91	53787	2354	5340
92	5105	249	0
93	595	13	0
94	7469	232	0
95	1411	40	0
96	2079	84	0
97	698	20	0
98	6584	396	0
99	5504	213	0

Table 15: Cluster Information and Union Metrics

Methods	TydiQA	MMLU	BBH	Average	OMP Time
BERT+Cluster+OMP	52.21	45.65	42.95	46.93	1h
Llama+Cluster+OMP	52.01	46.04	40.2	46.08	9.5h
Grad+Cluster+OMP (TAGCOS)	**52.78**	**48.01**	**44.26**	**48.35**	0.8h

Table 16: Performance Comparison of Different Methods

Metric	OMP Selected Data	reDistributed in Top 10 Categories
Entropy	5.467	3.322
Gini Coefficient	0.963	0.900
Coefficient of Variation	1.636	3.000

Table 17: OMP Selected Data Features.

Dataset	**Sourced from**	**# Instances**	**License**
SuperNI	NLP datasets + Human-written Instructions	96,913	Apache-2.0
CoT	NLP datasets + Human-written CoTs	100,000	ODC-BY
Flan V2	NLP datasets + Human-written Instructions	100,000	Apache-2.0
Dolly	Human-written from scratch	15,011	Apache-2.0
Self-instruct	Generated w/ vanilla GPT3 LM	82,439	Apache-2.0
Unnatural Instructions	Generated w/ Davinci-002	68,478	MIT
Code-Alpaca	Generated w/ Davinci-003	20,022	Apache-2.0
GPT4-Alpaca	Generated w/ Davinci-003+GPT4	52,002	Apache-2.0
Baize	Generated w/ ChatGPT	210,311	GPL-3.0
ShareGPT	User prompts + outputs from various models	168,864	Apache-2.0
WizardLM	Generated w/ GPT-3.5-Turbo	30,000	-
Oasst1	Human-written from scratch	33,919	Apache-2.0
Hardcoded	-	14	ODC-BY
LIMA	Human-written from scratch	1,030	CC-BY-NC-SA
Science Literature	NLP datasets	7,544	ODC-BY
Open-Orca	Generated w/ GPT4	30,000	MIT
Standford Alpaca	Generated w/ Davinci-003	52,002	Apache-2.0

Table 18: Details of datasets used in our paper.