LESS: Selecting Influential Data for Targeted Instruction Tuning

Ziyan Wang, Mehul Kalia

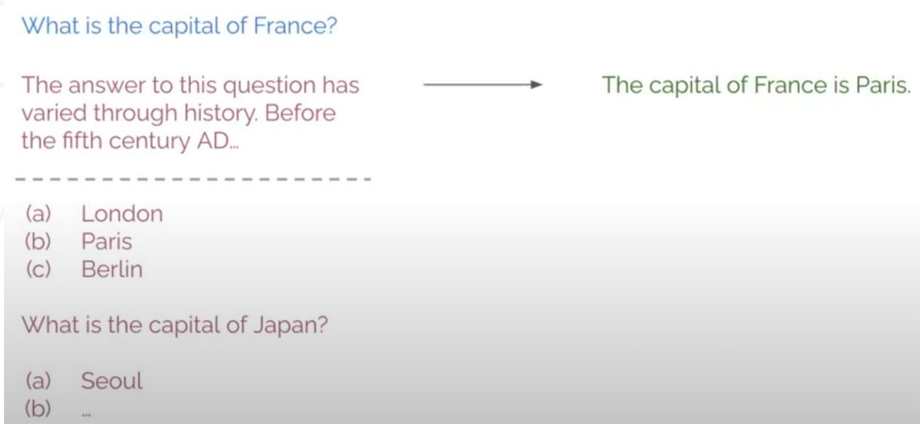
Georgia Tech

Outline

- Introduction
- Preliminary
- Method
- ==========Separation Line==========
- Experiment Setup
- Results

Introduction

Instruction tuning teaches a model to follow instructions, allowing it to perform new, *unseen* tasks.

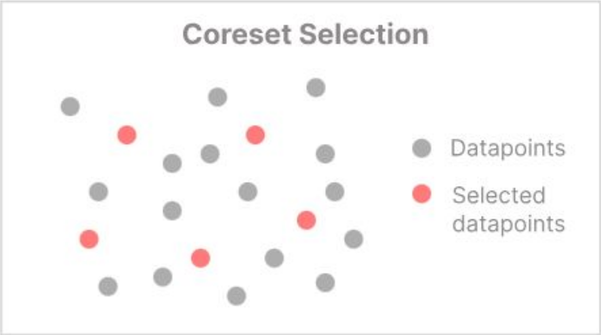


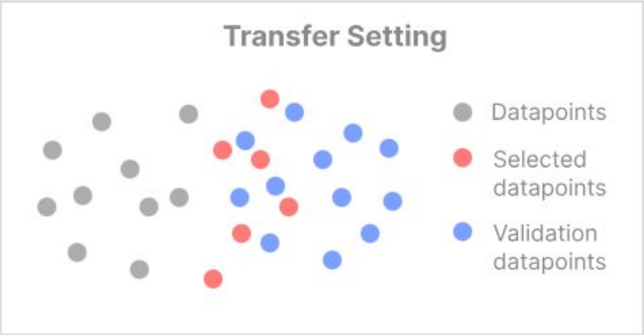
Many real-world applications call for cultivating a specific suite of capabilities in LLMs (e.g., reasoning skills). However, training LLMs with mixed instruction tuning datasets can hinder the development of these specific capabilities. For example, Wang et al. (2023b) demonstrates that LLMs trained on a mix of instruction tuning datasets exhibit worse performance than those trained on a subset of the data. Additionally, considering the broad spectrum of user queries and the multitude of skills required to respond to them, there may not always be enough in-domain data available. Therefore, we hope to be able to effectively use the general instruction tuning data to improve specific capabilities. We frame this setting as *targeted instruction tuning*:

Given just a handful of examples embodying a specific capability, how can we effectively select relevant fine-tuning data from a large collection of instruction datasets?

Firgure from @https://www.youtube.com/watch?v=YoVek79LFe0

Introduction





Coreset selection **selects data such that the selected subset represents the full dataset.**

Transfer data selection **selects the subset that is closest to the target data points.**

Firgure from @https://www.cs.princeton.edu/~smalladi/blog/2024/04/04/dataselection/

Preliminary

The first-order Taylor s
by the formula:

$$f(x) \approx f(a) + f'(a)($$

Per-step influence. Consider a model $\boldsymbol{\theta}^t$ at time step t trained on the loss $\ell(\cdot;\boldsymbol{\theta}^t)$. We can write the first-order Taylor expansion of the loss on a validation datapoint $\boldsymbol{z}'$ as

$$\ell(\boldsymbol{z}';\boldsymbol{\theta}^{t+1}) \approx \ell(\boldsymbol{z}';\boldsymbol{\theta}^t) + \langle \nabla \ell(\boldsymbol{z}';\boldsymbol{\theta}^t), \boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^t \rangle$$

For ease of exposition, assume that we are training the model with SGD with batch size 1 and learning rate η_t.[2] If $\boldsymbol{z}$ is the training data at time step t, we can write the SGD update as $\boldsymbol{\theta}^{t+1} - \boldsymbol{\theta}^t = -\eta_t \nabla \ell(\boldsymbol{z};\boldsymbol{\theta}^t)$. Then, the Taylor expansion can be written as

$$\ell(\boldsymbol{z}';\boldsymbol{\theta}^{t+1}) - \ell(\boldsymbol{z}';\boldsymbol{\theta}^t) \approx -\eta_t \langle \nabla \ell(\boldsymbol{z};\boldsymbol{\theta}^t), \nabla \ell(\boldsymbol{z}';\boldsymbol{\theta}^t) \rangle$$

Trajectory influence. The influence of $\boldsymbol{z}$ over the entire training run can be measured by aggregating the influence at every training step that uses $\boldsymbol{z}$. Since $\boldsymbol{z}$ is used once per epoch, it is natural to express this as a summation over epochs:

$$\mathrm{Inf}_{\mathrm{SGD}}(\boldsymbol{z}, \boldsymbol{z}') \triangleq \sum_{i=1}^{N} \bar{\eta}_i \langle \nabla \ell(\boldsymbol{z}';\boldsymbol{\theta}_i), \nabla \ell(\boldsymbol{z};\boldsymbol{\theta}_i) \rangle \quad (1)$$

where $\bar{\eta}_i$ is the learning rate used during the ith epoch out of N total training epochs and $\boldsymbol{\theta}_i$ is the model after the ith epoch of training.

Preliminary

Data selection with influence. While Pruthi et al. (2020) used this insight to identify mislabeled training data, we instead apply this formula to design a data selection strategy. In particular, at each time step t, selecting $\boldsymbol{z}$ to maximize $\langle \nabla \ell(\boldsymbol{z}'; \boldsymbol{\theta}^t), \nabla \ell(\boldsymbol{z}; \boldsymbol{\theta}^t) \rangle$ will drive a larger decrease in the loss on the validation point $\boldsymbol{z}'$. However, when computing $\mathrm{Inf}_{\mathrm{SGD}}$ across several epochs, we note that the model checkpoints $\{\boldsymbol{\theta}_i\}$ after the first epoch will depend on the dataset selected for training. This causes the data selection problem to become circular, and we empirically circumvent this problem with a short *warmup training* run on a randomly selected $\mathcal{D}_{\mathrm{warmup}} \subset \mathcal{D}$ for $N = 4$ epochs (see §4.1). Overall, this data selection strategy is especially useful in the transfer learning setting, because it does not require any specific relationship between $\boldsymbol{z}'$ and $\boldsymbol{z}$. The next two sections describe how we adapt this basic approach to operate efficiently and effectively with instruction tuning.

Trajectory influence. The influence of $\boldsymbol{z}$ over the entire training run can be measured by aggregating the influence at every training step that uses $\boldsymbol{z}$. Since $\boldsymbol{z}$ is used once per epoch, it is natural to express this as a summation over epochs:

$$\mathrm{Inf}_{\mathrm{SGD}}(\boldsymbol{z}, \boldsymbol{z}') \triangleq \sum_{i=1}^{N} \bar{\eta}_i \langle \nabla \ell(\boldsymbol{z}'; \boldsymbol{\theta}_i), \nabla \ell(\boldsymbol{z}; \boldsymbol{\theta}_i) \rangle \quad (1)$$

where $\bar{\eta}_i$ is the learning rate used during the ith epoch out of N total training epochs and $\boldsymbol{\theta}_i$ is the model after the ith epoch of training.

Preliminary

3.1. Extension to Adam

The formulation in Equation (1) is unique to optimizing models with SGD. However, instruction tuning is usually performed using the Adam optimizer (Kingma & Ba, 2015).[4] In this case, the parameter update at a given step is:

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

$$\boxed{\Gamma(z, \theta^t)} \triangleq \frac{m^{t+1}}{\sqrt{v^{t+1}} + \epsilon}$$

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

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

where all operations are performed elementwise, with β_1 and β_2 as the hyperparameters for the first and second moments, respectively, and ϵ as a small constant. Then, the first-order expansion for the Adam dynamics suggests we should choose z to maximize $\langle \nabla \ell(z'; \theta^t), \Gamma(z, \theta^t) \rangle$. Note that extending the data selection strategy to Adam exacerbates the aforementioned circularity of the procedure, because computing $\Gamma(z, \theta)$ requires accessing the m and v terms, which are determined by prior training gradients. As before, we obtain these from the warmup training (§4.1).[5]

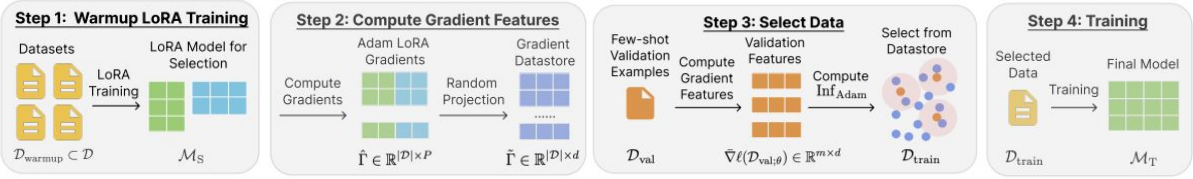
Trajectory influence. The influence of z over the entire training run can be measured by aggregating the influence at every training step that uses z. Since z is used once per epoch, it is natural to express this as a summation over epochs:

$$\mathrm{Inf}_{\mathrm{SGD}}(z, z') \triangleq \sum_{i=1}^{N} \bar{\eta}_i \langle \nabla \ell(z'; \theta_i), \boxed{\nabla \ell(z; \theta_i)} \rangle \quad (1)$$

where $\bar{\eta}_i$ is the learning rate used during the ith epoch out of N total training epochs and θ_i is the model after the ith epoch of training.

Georgia Tech

Method Overview

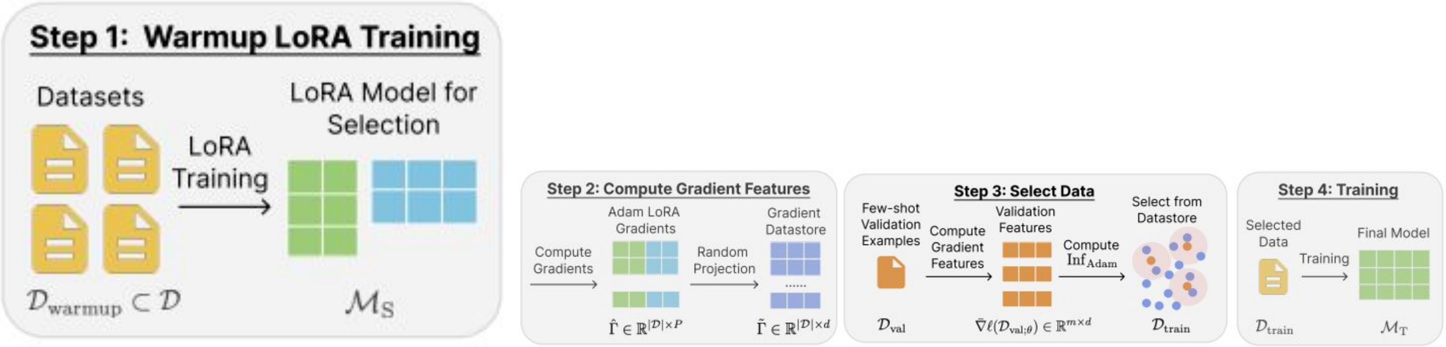


Delve into Method

Step 1

During step 1, we compute $\hat{\Gamma}(\cdot, \boldsymbol{\theta})$

Step 1: Warmup training with LoRA. We use LoRA (Hu et al., 2021) to reduce the number of trainable parameters and accelerate the inner products in Definition 3.1. LoRA freezes the pre-trained weights and adds a low-rank adaptor to linear layers throughout the network. We use LoRA to instruction tune a pre-trained base model (e.g., LLAMA-2-7B) on a random subset $\mathcal{D}_{\text{warmup}} \subset \mathcal{D}$ for N epochs (we only use 5% of the training data in practice, see §5.1), checkpointing the model after each epoch to store $\{\boldsymbol{\theta}_i\}_{i=1}^{N}$. The gradient when training with LoRA, denoted $\hat{\nabla}\ell(\cdot;\boldsymbol{\theta}) \in \mathbb{R}^P$, is much lower dimensional than the model itself; for example, in LLAMA-2-7B, $\hat{\nabla}\ell(\cdot;\boldsymbol{\theta})$ is less than 2% the size of $\boldsymbol{\theta}$. We use $\hat{\nabla}\ell(\cdot;\boldsymbol{\theta})$ to compute the Adam update and denote it as $\hat{\Gamma}(\cdot, \boldsymbol{\theta})$. This initial warmup training is motivated conceptually in §3.1, and empirical results in §6.1 demonstrate that omitting it yields suboptimal results.



Delve into Method

Step 2

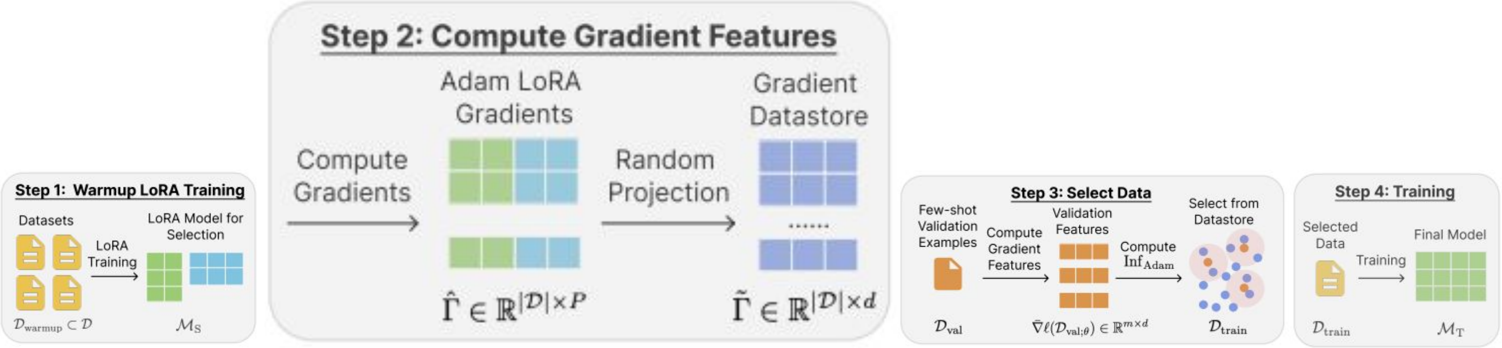
During step 2, we compute

$$\tilde{\nabla}\ell(\boldsymbol{z}'; \boldsymbol{\theta}_i)$$

$$\tilde{\Gamma}(\boldsymbol{z}, \cdot)$$

Step 2: Projecting the gradients. To further reduce the feature dimensionality, we apply a random projection to the LoRA gradients. The Johnson-Lindenstrauss Lemma (Johnson & Lindenstrauss, 1984) asserts that such projections often preserve the inner products in Definition 3.1, thereby ensuring these low-dimensional gradient features are still useful for dataset selection. For a given validation datapoint $\boldsymbol{z}'$ and model checkpoint $\boldsymbol{\theta}_i$, we can compute a d-dimensional projection of the LoRA gradient $\tilde{\nabla}\ell(\boldsymbol{z}'; \boldsymbol{\theta}_i) = \Pi^\top \hat{\nabla}\ell(\boldsymbol{z}'; \boldsymbol{\theta}_i)$, with each entry of $\Pi \in \mathbb{R}^{P \times d}$ drawn from a Rademacher distribution (i.e., $\Pi_{ij} \sim \mathcal{U}(\{-1, 1\})$). For training datapoints $\boldsymbol{z}$, we compute $\tilde{\Gamma}(\boldsymbol{z}, \cdot) = \Pi^\top \hat{\Gamma}(\boldsymbol{z}, \cdot)$.

We use the memory-efficient online implementation of random projections from Park et al. (2023) to compute and apply Π. In practice, we choose $d = 8192$.



Georgia Tech

Delve into Method

Step 3

Definition 3.1 (Adam Influence). Suppose the model is trained for N epochs, where $\bar{\eta}_i$ is the average learning rate in the ith epoch and $\boldsymbol{\theta}_i$ is the model checkpoint after the ith epoch. We define the influence of a training datapoint $\boldsymbol{z}$ on a validation datapoint $\boldsymbol{z}'$ when training with Adam as

$$\mathrm{Inf}_{\mathrm{Adam}}(\boldsymbol{z}, \boldsymbol{z}') \triangleq \sum_{i=1}^{N} \bar{\eta}_i \cos(\nabla \ell(\boldsymbol{z}'; \boldsymbol{\theta}_i), \Gamma(\boldsymbol{z}, \boldsymbol{\theta}_i))$$

where cos computes the cosine similarity of the two vectors.

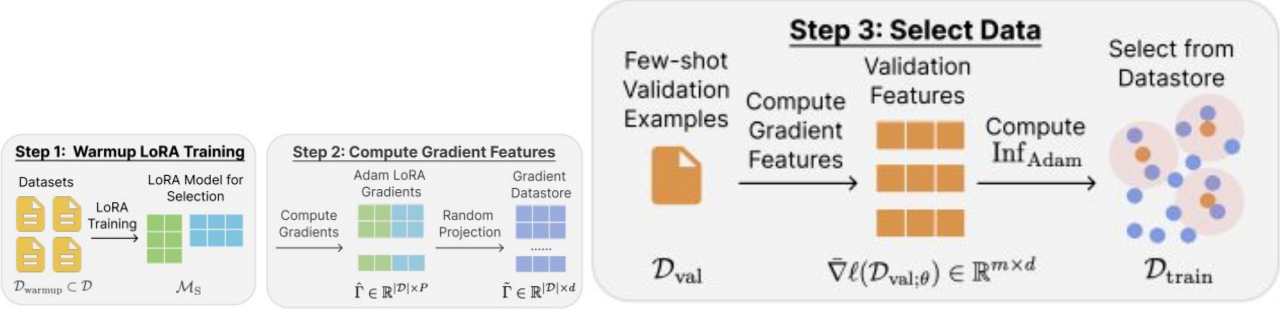
Trajectory influence. The influence of $\boldsymbol{z}$ over the entire training run can be measured by aggregating the influence at every training step that uses $\boldsymbol{z}$. Since $\boldsymbol{z}$ is used once per epoch, it is natural to express this as a summation over epochs:

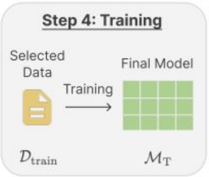
$$\mathrm{Inf}_{\mathrm{SGD}}(\boldsymbol{z}, \boldsymbol{z}') \triangleq \sum_{i=1}^{N} \bar{\eta}_i \langle \nabla \ell(\boldsymbol{z}'; \boldsymbol{\theta}_i), \overset{\boxed{\Gamma(\boldsymbol{z}, \boldsymbol{\theta}^t)}}{\cancel{\nabla \ell(\boldsymbol{z}; \boldsymbol{\theta}_i)}} \rangle \quad (1)$$

where $\bar{\eta}_i$ is the learning rate used during the ith epoch out of N total training epochs and $\boldsymbol{\theta}_i$ is the model after the ith epoch of training.

$$\mathrm{Inf}_{\mathrm{Adam}}(\boldsymbol{z}, \mathcal{D}_{\mathrm{val}}^{(j)}) = \sum_{i=1}^{N} \bar{\eta}_i \frac{\langle \bar{\nabla} \ell(\mathcal{D}_{\mathrm{val}}^{(j)}; \boldsymbol{\theta}_i), \Gamma(\boldsymbol{z}, \boldsymbol{\theta}_i) \rangle}{\|\bar{\nabla} \ell(\mathcal{D}_{\mathrm{val}}^{(j)}; \boldsymbol{\theta}_i)\| \|\tilde{\Gamma}(\boldsymbol{z}, \boldsymbol{\theta}_i)\|}. \quad (2)$$

We select training datapoints that can improve performance on any one of the validation subtasks. Following the logic in §2, we compute the score for $\boldsymbol{z}$ as the maximum across all subtasks: $\max_j \mathrm{Inf}_{\mathrm{Adam}}(\boldsymbol{z}, \mathcal{D}_{\mathrm{val}}^{(j)})$. We select the highest scoring examples to construct $\mathcal{D}_{\mathrm{train}}$.[7] After selection, we use the selected subset $\mathcal{D}_{\mathrm{train}}$ to train the target model $\mathcal{M}_T$.





Method Revisit

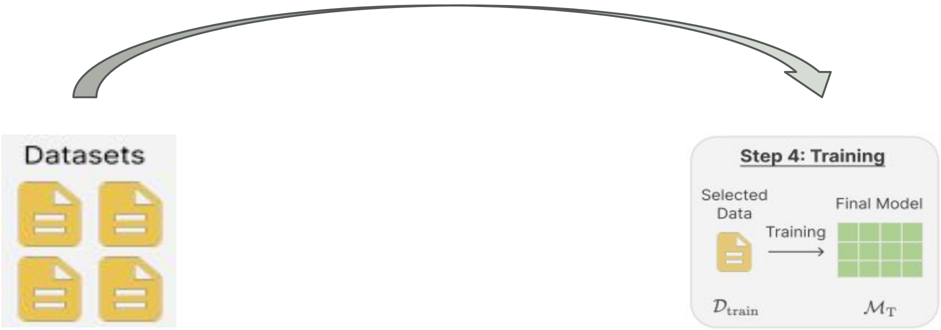
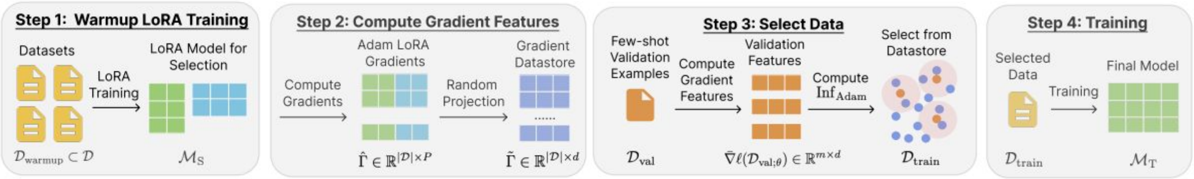


Figure 1: Illustration of LESS. In step 1, we train a selection model $\mathcal{M}_S$ with LoRA for a warmup period with a small subset of data $\mathcal{D}_{\text{warmup}} \subset \mathcal{D}$. In step 2, we compute the Adam LoRA gradient features $\Gamma \in \mathbb{R}^{|\mathcal{D}| \times P}$ for each candidate datapoint and save them in a gradient datastore. In step 3, for any task with few-shot examples $\mathcal{D}_{\text{val}}$ (comprising of m subtasks), we compute the gradient features for each validation subtask and select the subset $\mathcal{D}_{\text{train}}$ with the top 5% training examples ranked by Inf_{Adam}. Step 4 is the final training stage with the selected data on a target model $\mathcal{M}_T$, which can be trained with either LoRA or full finetuning. Steps 1 and 2 are offline and only need to be computed once per candidate training set $\mathcal{D}$.



Separation slide

Experimental Setup

- Instruction Tuning Datasets
- Evaluation Datasets
- Models Used
- Selection and Training Procedure

Datasets Used for Instruction Tuning

Dataset	# Instance	Sourced from	# Rounds	Prompt Len.	Completion Len.
FLAN V2	100,000	NLP datasets and human-written instructions	1	355.7	31.2
CoT	100,000	NLP datasets and human-written CoTs	1	266	53.2
DOLLY	15,011	Human-written from scratch	1	118.1	91.3
OPEN ASSISTANT 1	55,668	Human-written from scratch	1.6	34.8	212.5

- LESS used a mix of instruction datasets covering a variety of tasks and reasoning, ~ 270k total points
- **No obvious in-domain data** for target queries included here

Evaluation Datasets

Dataset	# Shot	# Tasks	$\|\mathcal{D}_{\text{val}}\|$	$\|\mathcal{D}_{\text{test}}\|$	Answer Type
MMLU	5	57	285	18,721	Letter options
TYDIQA	1	9	9	1,713	Span
BBH	3	23	69	920	COT and answer

- 3 different datasets used to simulate real-world instruction tuning needs

Models Used

- Models used
  - Llama-2 (7B and 13B)
  - Mistral-7B
- LESS-T
  - Transfer setting
  - Tests data selection efficiency by using a smaller model (LLAMA-2-7B) to select data for larger models.
- Warmup training on 5% of full dataset
- Warmup and final training conducted w/ LoRA

Selection and Training Procedure

- Warmup LoRA Training of randomly selected 5% of data
- Compute Gradient Features (Construct gradient datastore)
- Score Datapoints
- Final Training on Top 5% Scored Data

Results

- Performance Metrics
- Comparison w/ Baselines
- Transferability
- Efficiency and Interpretability
- Qualitative Analysis

Results- Performance Metrics

Data percentage	MMLU Full (100%)	MMLU Rand. (5%)	MMLU LESS-T (5%)	MMLU LESS (5%)	TYDIQA Full (100%)	TYDIQA Rand. (5%)	TYDIQA LESS-T (5%)	TYDIQA LESS (5%)	BBH Full (100%)	BBH Rand. (5%)	BBH LESS-T (5%)	BBH LESS (5%)
LLAMA-2-7B	*51.6*	46.5 (0.5)	-	**50.2** (0.5)	*54.0*	52.7 (0.4)	-	**56.2** (0.7)	*43.2*	38.9 (0.5)	-	**41.5** (0.6)
LLAMA-2-13B	*54.5*	53.4 (0.1)	**54.6** (0.3)	54.0 (0.7)	*54.3*	53.0 (1.3)	**57.5** (0.8)	54.6 (0.3)	*50.8*	47.0 (1.6)	49.9 (0.5)	**50.6** (0.6)
MISTRAL-7B	*60.4*	60.0 (0.1)	60.6 (0.3)	**61.8** (0.4)	*57.7*	56.9 (0.2)	**61.7** (1.7)	60.3 (2.4)	*53.0*	54.5 (0.1)	**56.0** (0.8)	**56.0** (1.0)

- LESS beats random selection for all models and tasks, does well on challenging tasks like TYDIQA and BBH
- Underlined numbers show when LESS beats using the full dataset
  - Filters out irrelevant data

Results- Comparison w/ Baselines

	Rand.	BM25	DSIR	RDS	LESS	Δ
MMLU	46.5 (0.5)	47.6	46.1 (0.3)	45.0 (1.0)	**50.2** (0.5)	↑2.6
TYDIQA	52.7 (0.4)	52.7	44.5 (1.7)	46.8 (1.3)	**56.2** (0.7)	↑3.5
BBH	38.9 (0.5)	39.8	36.8 (0.1)	36.7 (1.3)	**41.5** (0.6)	↑1.7

- Baselines Used (5% used for each)
    - Random selection
    - **B**est **M**atching **25** (word frequency stats for data ranking)
    - DSIR (n-gram feature weighting)
    - RDS (model-based representation features)
- LESS beats all baselines

Results- Transferability

	MMLU				TYDIQA				BBH			
Data percentage	Full (100%)	Rand. (5%)	LESS-T (5%)	LESS (5%)	Full (100%)	Rand. (5%)	LESS-T (5%)	LESS (5%)	Full (100%)	Rand. (5%)	LESS-T (5%)	LESS (5%)
LLAMA-2-7B	*51.6*	46.5 (0.5)	-	**50.2** (0.5)	*54.0*	52.7 (0.4)	-	**56.2** (0.7)	*43.2*	38.9 (0.5)	-	**41.5** (0.6)
LLAMA-2-13B	*54.5*	53.4 (0.1)	**54.6** (0.3)	54.0 (0.7)	*54.3*	53.0 (1.3)	**57.5** (0.8)	54.6 (0.3)	*50.8*	47.0 (1.6)	49.9 (0.5)	**50.6** (0.6)
MISTRAL-7B	*60.4*	60.0 (0.1)	60.6 (0.3)	**61.8** (0.4)	*57.7*	56.9 (0.2)	**61.7** (1.7)	60.3 (2.4)	*53.0*	54.5 (0.1)	**56.0** (0.8)	**56.0** (1.0)

- Works well across three models tested
- LESS-T
  - Uses smaller model gradients for selection
  - Does comparable to, and sometimes beats, full-model selection (LESS)
  - Saves computational resources- gradient features calculated on smaller scale but generalizes well
  - **Takeaway- small models can effectively select data for other models in pre-training**

Results- Efficiency

	Warmup LoRA Training		Gradient Features Computation		Data Selection	
	Complexity	**Actual**	**Complexity**	**Actual**	**Complexity**	**Actual**
Compute	$\mathcal{O}(\lvert\mathcal{D}_{\text{warmup}}\rvert \cdot N)$	6 Hours	$\mathcal{O}(\lvert\mathcal{D}\rvert \cdot N)$	48 Hours	$\mathcal{O}(\lvert\mathcal{D}\rvert \cdot \lvert\mathcal{D}_{\text{val}}\rvert \cdot d)$	$<$ 1 Min
Storage	-	-	$\mathcal{O}(\lvert\mathcal{D}\rvert \cdot N \cdot d)$	17.7 GB	-	-

- **N:** Number of epochs, **|D|**: Dataset size, **d:** Projected Gradient Dimension
- Gradient store (bottleneck) is reusable across tasks without recomputation
    - Only needs to be done once per dataset

Qualitative Results- Interpretability

A TydiQA Validation Example: Question Answering in Bengali (Translated)

User: Answer the following question based on the content of the given chapter.
Chapter: The Bengali Renaissance and the Brahmo Samaj - in a nutshell, the social reformers and the reforms they introduced had a profound impact on the social and economic life of Bengal. The beginning of the great rebellion took place in Calcutta in 1857. After the failure of this rebellion, the British Empire accepted the rule of the East India Company's hands as a friendly power. For the governance of India, the position of a Viceroy was created. In 1905, the religious and political motivations led to the division of Bengal [...]
Question: When was Bengal divided?

Assistant: Answer: 1905.

Selected by BM25

Masked Word Prediction in Bengali

User: Select the most logical word from four options to replace the <MASK> token in the given Bengali statement.
[Q]: Statement: ১৯৬১-৬২ মৌসুমে টেড ডেক্সটারের নেতৃত্বাধীন ইংরেজ দলের সদস্যরূপে পাকিস্তান ও <MASK> সফরে ঐ দুই টেস্টে অংশগ্রহণ করেন। তার বোলিং ভঙ্গীমার কারণে টেস্ট অভিষেকে পর্ব ক্ষাণিকটা বিলম্বিত হয়। Option A: গ্ল্যামারগন Option B: লিচেস্টারশাযারের Option C: ভারত

Assistant: [A]: ভারত

Selected by RDS

Hate Speech Classification in Bengali

User: You are given a hateful post in Bengali that expresses hate or encourages violence towards a person or a group based on the protected characteristics such as race, religion, sex, and sexual orientation. You are expected to classify the post into two classes: personal or non-personal depending on the topic.
Q: তুমি কি সুখী হতে চাও না? না চাই না কিন্তু কেন? সুখে থাকলে ভূতে কিলায় আমি কিল খেতে চাই না

Assistant: personal

Selected by LESS

Question Answering in English

User: Given the question and input, write a reponse to answer the question. Which year was quantum computer demonstrated to be possible?
Input: Over the years, experimentalists have constructed small-scale quantum computers using trapped ions and superconductors. In 1998, a two-qubit quantum computer demonstrated the feasibility of the technology, [...]
Response:

Assistant: 1998

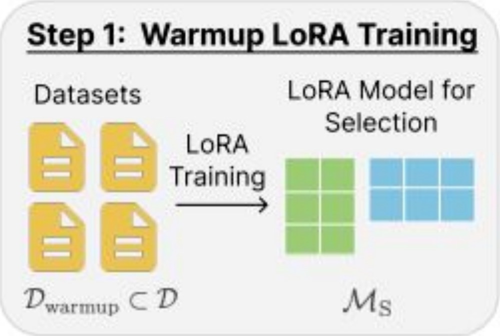
Georgia Tech

Qualitative Results- Interpretability

Selected by BM25	Selected by RDS	Selected by LESS
Masked Word Prediction in Bengali **User:** Select the most logical word from four options to replace the <MASK> token in the given Bengali statement. [Q]: Statement: ১৯৬১-৬২ মৌসুমে টেড ডেক্সটারের নেতৃত্বাধীন ইংরেজ দলের সদস্যরূপে পাকিস্তান ও <MASK> সফরে ঐ দুই টেস্টে অংশগ্রহণ করেন। তার বোলিং ভঙ্গীমার কারণে টেস্ট অভিষেকে পর্ব ক্ষাণিকটা বিলম্বিত হয়। Option A: গ্ল্যামারগন Option B: লিচেস্টারশাযারের Option C: ভারত **Assistant:** [A]: ভারত	*Hate Speech Classification in Bengali* **User:** You are given a hateful post in Bengali that expresses hate or encourages violence towards a person or a group based on the protected characteristics such as race, religion, sex, and sexual orientation. You are expected to classify the post into two classes: personal or non-personal depending on the topic. Q: তুমি কি সুখী হতে চাও না? না চাই না কিন্তু কেন? সুখে থাকলে ভূতে কিলায় আমি কিল খেতে চাই না **Assistant:** personal	*Question Answering in English* **User:** Given the question and input, write a reponse to answer the question. Which year was quantum computer demonstrated to be possible? Input: Over the years, experimentalists have constructed small-scale quantum computers using trapped ions and superconductors. In 1998, a two-qubit quantum computer demonstrated the feasibility of the technology, [...] Response: **Assistant:** 1998

- Interpretability
  - LESS selects examples based on reasoning similarity, not superficial similarities
- In TYDIQA (multi-lingual), LESS selects English examples that match the task despite different language

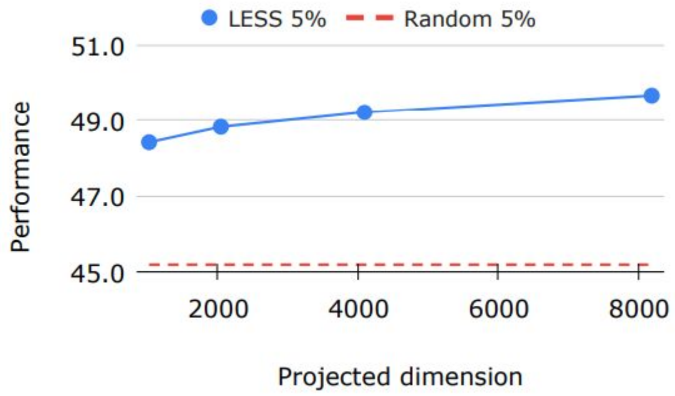
Results- Is Warmup Training Neccessary



	LLAMA-2-7B		LoRA Models		
	Base (0%)	**Chat** (Unk.)	**5%** (default)	**25%**	**100%**
MMLU	46.7	47.9	50.2	51.3	51.6
TYDIQA	52.1	52.2	56.2	57.0	57.9
BBH	39.8	38.6	41.5	41.5	41.9
Avg.	46.2	46.2	49.3	49.9	**50.5**

- Left: Vanilla gradients (no warmup training), Right: Gradients from LoRA models
- Performance increases with size of |Dwarmup|

Results- Varying Gradient Projection Dimension



	Random	Projected Gradient Dimension 1024	2048	4096	8192
MMLU	46.5	50.7	**51.2**	50.5	51.1
TydiQA	52.7	55.3	56.3	**56.8**	56.6
BBH	38.9	39.3	39.0	40.4	**41.3**
Average	45.2	48.4	48.8	49.2	**49.7**

- Left: Average of three datasets, Right: Dataset Breakdown
- On average, increasing gradient projection dimension improves performance
- This comes at a larger computational cost

Takeaways

- LESS's targeted data selection for instruction tuning achieves competitive performance with only 5% of training data
- Gradient similarity approach performs well across tasks and model sizes

Thanks