Threshold Filtering Packing for Supervised Fine-Tuning: Training Related Samples within Packs

Jiancheng Dong[1], **Lei Jiang**[1], **Wei Jin**[2], **Lu Cheng**[1]
[1]University of Illinois Chicago, [2]Emory University
dongjiancheng77@gmail.com, {ljian43, lucheng}@uic.edu, wei.jin@emory.edu

Abstract

Packing for Supervised Fine-Tuning (SFT) in autoregressive models involves concatenating data points of varying lengths until reaching the designed maximum length to facilitate GPU processing. However, randomly concatenating data points can lead to cross-contamination of sequences due to the significant difference in their subject matter. The mainstream approaches in SFT ensure that each token in the attention calculation phase only focuses on tokens within its own short sequence, without providing additional learning signals for the preceding context. To address these challenges, we introduce Threshold Filtering Packing (TFP), a method that selects samples with related context while maintaining sufficient diversity within the same pack. Our experiments show that TFP offers a simple-to-implement and scalable approach that significantly enhances SFT performance, with observed improvements of up to 7% on GSM8K, 4% on HumanEval. Furthermore, results from bias benchmark datasets highlight TFP's promising performance in improving fairness while also boosting prediction accuracy by 15%.

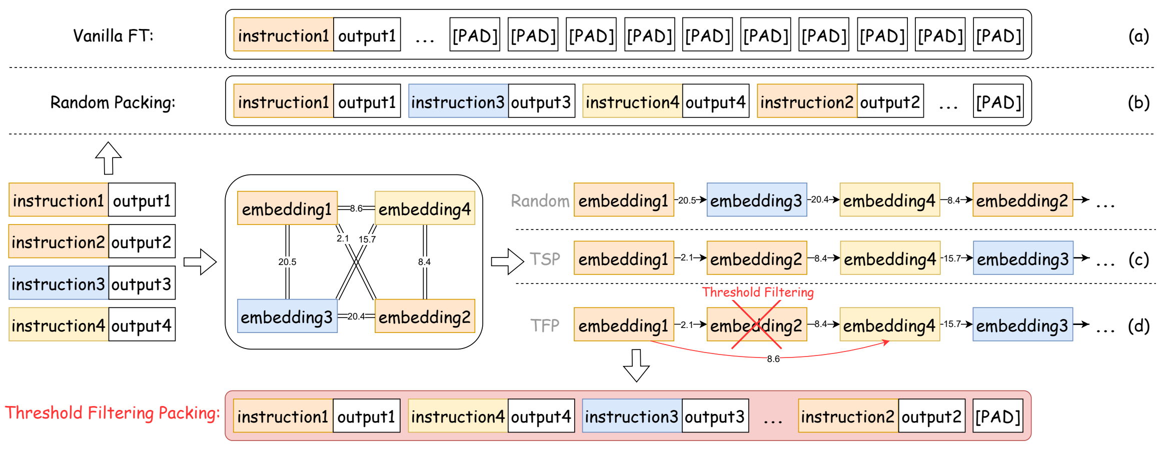
1 Introduction

In Supervised Fine-Tuning (SFT) for large language models (LLMs), sequence lengths can vary substantially, requiring the wrapping of data into tensors to apply matrix operations optimized for CUDA and GPUs (Raffel et al., 2020). As illustrated in Figure 1(a), vanilla fine-tuning pad shorter sequences with special tokens "[PAD]" up to the maximum sequence length. While this ensures uniformity, it introduces inefficiencies by including irrelevant padding tokens in the computation, wasting GPU resources and dilutes the model's learning signal (Kundu et al., 2024).

To address this issue, packing sequences has become a common technique in autoregressive transformer models during training and inference to optimize context length and reduce padding (Liu et al., 2019; Brown et al., 2020). This method involves randomly selecting and concatenating data of varying lengths until reaching the designed maximum length. Recent studies suggest that packed data, when batched and processed on multi-GPUs, effectively minimize idle time within each batch (Bai et al., 2024).

However, randomly concatenating these data samples (Figure 1(b)), can result in sequence cross-contamination (Krell et al., 2021). Cross-contamination occurs when predictions for one sequence are influenced by an unrelated sequence, complicating accurate predictions, especially when the subjects differ. For instance, imagine a model tasked with generating a multiplication table, followed by a prompt to list useful expressions in French. If these two sequences are concatenated without proper separation, the model might mix the tasks, producing results like "3 x 2 = Bonjour." This leads to incorrect and confusing outputs, where instructions and contexts become inappropriately blended. Moreover, current SFT pipelines (Kundu et al., 2024) cause previous samples to provide no signal for predicting the next sample, thereby reducing learning efficiency and negatively impacting the few-shot performance of LLMs.

To address these challenges, in this work, we present Threshold Filtering Packing (TFP), a new packing approach that packs sequences of related yet diverse samples, encouraging context richness and reasoning across sample boundaries. Specifically, we employ a greedy algorithm inspired by the Traveling Salesman Problem (TSP) (Applegate et al., 2006) to efficiently map out a path for segmentation into multiple packs. TFP further refine these packs by ensuring that overly similar samples are not grouped together. Setting the threshold is crucial in this context as it allows us to strike a balance between similarity and diversity within each pack, preventing homogeneity and ensuring



Vanilla FT:
instruction1 output1 ... [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD]
(a)
Random Packing:
instruction1 output1 instruction3 output3 instruction4 output4 instruction2 output2 ... [PAD]
(b)
instruction1 output1
instruction2 output2
instruction3 output3
instruction4 output4
embedding1
embedding4
embedding3
embedding2
8.6
2.1
15.7
20.5
8.4
20.4
Random embedding1 20.5 embedding3 20.4 embedding4 8.4 embedding2 ...
TSP embedding1 2.1 embedding2 8.4 embedding4 15.7 embedding3 ... (c)
Threshold Filtering
TFP embedding1 2.1 embedding2 8.4 embedding4 15.7 embedding3 ... (d)
8.6
Threshold Filtering Packing:
instruction1 output1 instruction4 output4 instruction3 output3 ... instruction2 output2 [PAD]

Figure 1: **Overview of TFP.** Different from vanilla FT (a), which uses "[PAD]" tokens up to the maximum length, and random packing (b), which places randomly shuffled samples in the same pack and may lead to cross-contamination, TFP (d) places related samples in the same pack while applying a threshold on TSP (c) to ensure that these samples are sufficiently distinct. This approach allows models to learn across diverse contexts and prevent overly similar samples from being grouped together.

the robustness of the generated packs. As shown in Figure 1(c), each sample is first converted to an embedding and then represented as a node in the graph. As shown in Figure 1(d), TFP employs threshold filtering to ensure each sample is distinct enough from recent ones, preventing the model from merely replicating previous outputs. This method results in packs that provide useful context while avoiding cross-contamination from unrelated texts.

For experiments, we fine-tune various LLMs on standard instruction fine-tuning datasets and conduct bias-related experiments to evaluate the potential effect of packing on fairness. Additionally, we assess the impact of TFP on computational efficiency for SFT. Our findings indicate that TFP demonstrates superior performance across various models. The bias-related experiments show that TFP offers a flexible operational space, allowing for adjustments in the ratio of sensitive attributes (e.g., race) within packs to effectively manage bias.

2 Related Work

SFT and Alignment Fine-tuning is a prevalent strategy to enhance model performance on downstream tasks, evidenced in domains such as coding (Wei et al., 2023; Luo et al., 2024) and arithmetic (Yue et al., 2024). Other work has highlighted the importance of consistency in format (Liang et al., 2023), data quality (Chung et al., 2022), and mixing tasks from different categories (Longpre et al., 2023; Iyer et al., 2022) in SFT. As LLMs evolve, the risk of generating unsafe content increases (Su et al., 2024; Wang et al., 2023a). Established methods for LLM alignment include instruction fine-tuning and reinforcement learning with human feedback (RLHF) (Ouyang et al., 2022). Instruction fine-tuning, also known as SFT, refines pre-trained models using annotated instructional data, often preceding RLHF to aid initial alignment (Touvron et al., 2023). RLHF employs reinforcement learning to adapt models based on feedback on generated responses. Although RLHF has been pivotal for developing systems like ChatGPT (OpenAI, 2021), isolated instruction fine-tuning can yield comparable outcomes (Sun et al., 2023) with much less computational and labor costs.

Packing While packing is relatively less researched, it is a technique extensively used in frameworks like Hugging Face's SFT Trainer [1] to expedite inference and training. To prevent cross-contamination during self-attention calculation, existing packing approaches involve concatenating sequences into a single tensor and using masking to disregard elements from other sequences during computation (Kundu et al., 2024). This method, including variations like LongAlign (Bai et al., 2024) and Prepacking (Zhao et al., 2024a), enhances training efficiency and minimizes the cross-contamination impact on model performance. However, it necessitates calculating a distinct attention mask for each batch, complicating implementation and increasing memory consumption for masks,

[1] `https://huggingface.co/docs/trl/sft_trainer`

which can hinder the effectiveness of flash attention.

In contrast, TFP avoids the need for masking to prevent cross-contamination. Instead, it forms data packs that provide relevant context without additional masking, streamlining implementation and reducing memory overhead.

3 Threshold Filtering Packing

The standard practice in packing is to form a pack by concatenating random samples until reaching the maximum context length (Zhao et al., 2024a). However, randomly concatenated packs do not provide additional learning signals and can lead to cross-contamination of sequences, compared to training on individual samples. In contrast, TFP generates more coherent packs by concatenating related and useful samples together, improving SFT performance and computational efficiency.

3.1 Problem Statement

Given a set of samples $D = \{d_i\}$, where each sample d_i has its instruction converted into an embedding e_i, our goal is to organize these samples into packs such that each of them consists of related samples that provide semantic context. Formally, we aim to form a set of packs $P_1 \cdots P_m$ where each pack $P_i = \{d_1, \dots, d_{k_i}\} \subset D$ and $\bigcup_{i=1}^{m} P_i = D$.

3.2 k-NN Packing

An intuitive method for packing is to apply k-NN and place each sample along with its retrieved top-k neighbors in the same pack, referred to as k-NN packing. This approach maintains sample similarity within each pack but introduces a significant issue: data repetition. Some samples frequently appear as nearest neighbors for multiple others, leading to overlapping packs, i.e., $\exists i \neq j, P_i \cap P_j \neq \emptyset$. For instance, in the CodeAlpaca dataset (Chaudhary, 2023), the sample "Construct a loop in Python to display all elements in a list" was included in 94 different packs with k-NN Packing, greatly reducing the diversity of pack content.

The data repetition problem can contaminate both individual packs and the entire training process. Within a pack, popular samples that are close to many others in the embedding space do not serve as diverse contexts, increasing the risk of cross-contamination. Across the training process, repeated exposure to these popular samples reduces the diversity of the dataset, potentially leading to overfitting (Shi et al., 2023).

3.3 Threshold Filtering Packing Algorithm

To address these challenges, we propose packing data samples that provide meaningful context while avoiding repeated selection. Recent research on pretraining language models with related documents has inspired our approach (Staniszewski et al., 2023; Shi et al., 2023; Zhao et al., 2024b), where similar documents are retrieved to enhance pretraining effectiveness. For Supervised Fine-Tuning datasets, we do not need to consider the degree of the starting node, as the shorter sequence length allows us to easily generate a complete graph. A basic approach involves using a greedy algorithm to select samples with the smallest Euclidean distance to their corresponding embeddings, ensuring that each sample is included only once. This is essentially a greedy algorithm for the TSP (Applegate et al., 2006). Intuitively, k-NN can select the same data point multiple times when retrieving the nearest neighbors, whereas TSP ensures that each data point is selected only once.

Further, previous studies (Yasunaga et al., 2023; Liu et al., 2023) show that maintaining diversity among the input contexts is crucial. We adopt a greedy algorithm for TSP with conditional adjustments and then segment this path into multiple packs to generate packs composed of diverse and relevant samples. TFP is designed to assemble related samples, with Threshold Filtering specifically addressing the challenge of placing overly similar samples in the same pack. As shown in Figure 1(d), threshold filtering separates overly similar embeddings, such as embedding1 and embedding2, which were initially connected by TSP.

The mathematical formulation of this approach is as follows:

$$\text{Minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{k_i} \sum_{l=j+1}^{k_i} w_{e_{ij},e_{il}}$$

$$\text{subject to} \quad \bigcup_{i=1}^{m} P_i = D,$$

$$P_i \cap P_j = \emptyset \quad \forall i \neq j,$$

$$|P_i| = k_i \quad \forall i,$$

$$w_{ij} = \sqrt{\sum_{q=1}^{n} (d_{i,q} - d_{j,q})^2},$$

$$w_{e_{ij},e_{il}} > t \quad \text{for all pairs of recent samples}$$

Here, m represents the number of packs. k_i is the number of elements in the i-th pack. e_{ij} refers to the j-th element in the i-th pack. w_{ij} is the Euclidean distance between elements e_i and e_j. $d_{i,k}$ represents the k-th feature of element e_i. n is the dimensionality of the feature space. t is the distance threshold to ensure diversity within a pack.

This objective function minimizes the sum of the Euclidean distances $w_{e_{ij},e_{il}}$ between all pairs of elements within each pack P_i. These constraints guarantee that the packs are disjoint, meaning no two packs share any common elements, and ensure that each pack P_i contains exactly k_i elements. The Euclidean distance between elements e_i and e_j is calculated based on their feature vectors $d_{i,k}$. Additionally, to ensure diversity within each pack, the constraint $w_{e_{ij},e_{il}} > t$ is applied . This method results in packs that provide useful context while avoiding cross-contamination from unrelated texts.

Algorithm 1 Threshold Filtering Packing

```
1: Input: Instruction embeddings array $E$, distance threshold $t$, the distance threshold number $r$
2: Output: A shortest path satisfying the threshold condition
3: Initialize $n \leftarrow$ length of $E$, $visited \leftarrow$ [False] $\times\ n$ , $visited[0] \leftarrow$ True
4: for $i \in \{1, 2, \ldots, n-1\}$ do
5:     $next \leftarrow$ SelectNext($E$, $visited$, $t$, $r$)
6:     // SelectNext($E$, $visited$, $t$, $r$) select the nearest unvisited embedding in $E$ while ensuring the distance from the recent $r$ samples is greater than $t$
7:     $path \leftarrow path \cup [next]$
8:     $visited[next] \leftarrow$ True
9: end for
10: return $path$
```

As depicted in Algorithm 1, TFP consists of three primary steps: first, generating sentence embeddings E for the instruction parts of the samples, then selecting the nearest unvisited embedding in E while ensuring that the distance from the recent r samples is greater than distance threshold t. Subsequently, we use the pack formed by instruction-related samples to train the language model. Since TFP only changes the distribution of data within the pack, it can be seamlessly integrated into existing SFT pipelines for LLMs. As a final step, we traverse the samples along the path and concatenate them to create packs.

4 Experiment

We evaluate the proposed approach from three aspects: (1) *Performance Comparison*: We compare TFP with various LLMs fine-tuned on common instruction fine-tuning datasets under both zero-shot and few-shot settings. (2) *Bias and Fairness*: Inspired by previous research (Wang et al., 2023a) that studies the impact of the ratio of different demographic groups in in-context learning (ICL) on LLM fairness, we investigate how adjusting the ratio in each pack during SFT can influence the bias and fairness of LLMs. (3) *Efficiency*: We study how various SFT methods influence computational efficiency on different GPU setups.

4.1 Experimental Setup

Datasets. We use commonly adopted datasets for instruction fine-tuning, which include tasks related to helpfulness, code-generation capabilities, and mathematical reasoning: (1) Alpaca dataset, generated from the Self-Instruct method (Wang et al., 2023c) via the text-davinci-003 model (Buruk, 2023), covering various tasks such as arithmetic, coding, and question-answering. (2) CodeAlpaca dataset(Chaudhary, 2023), which aims to build and share an instruction-following LLaMA model for code generation. (3) GSM8K dataset (Cobbe et al., 2021), curated to examine mathematical reasoning capabilities, comprises 8.8k high-quality arithmetic word problems designed at the grade school level.

For the fairness-related experiments, we use the Jigsaw Unintended Bias in Toxicity Classification task (Adams et al., 2019) and Adult dataset (Becker and Kohavi, 1996). The Jigsaw Unintended Bias in Toxicity Classification task involves performing toxicity classification on comment texts published by the Civil Comments platform. It contains human-annotated demographic information such as race, gender, and religion. The goal is to ensure that models make predictions based on the text toxicity, rather than demographic information included in the text. We use race (Black and non-Black) as the protected attributes. The Adult dataset is a tabular dataset that includes 14 attributes of a person (e.g., age and education level) as input, to predict whether the person's income exceeds $50k per year. We evaluate the fairness of fine-tuned models based on the sensitive attribute of sex, specifically com-

	Llama2-7B			Llama3-8B			Mistral-7B		
Method	**WR**	**HumanEval**	**GSM8K**	**WR**	**HumanEval**	**GSM8K**	**WR**	**HumanEval**	**GSM8K**
Vanilla FT	48.2 ± 0.4	19.5 ± 0.3	26.2 ± 0.0	51.2 ± 0.5	38.4 ± 0.0	61.8 ± 0.2	62.9 ± 0.6	35.4 ± 0.3	59.7 ± 0.5
Sorted batching	48.2 ± 0.5	20.1 ± 0.3	26.5 ± 0.0	51.8 ± 0.6	37.2 ± 0.4	62.0 ± 0.6	61.2 ± 0.3	34.1 ± 0.5	61.0 ± 0.1
Random packing	47.1 ± 0.3	19.5 ± 0.3	26.1 ± 0.4	51.7 ± 0.5	37.8 ± 0.3	62.1 ± 0.6	62.4 ± 0.0	34.8 ± 0.6	59.1 ± 0.4
Random packing (mask)	47.6 ± 0.5	19.5 ± 0.6	26.1 ± 0.2	52.4 ± 0.0	37.8 ± 0.3	62.5 ± 0.4	**63.5 ± 0.2**	35.4 ± 0.5	59.2 ± 0.1
Packing+loss weighting	47.1 ± 0.6	18.9 ± 0.4	25.8 ± 0.0	51.2 ± 0.3	38.4 ± 0.0	60.9 ± 0.3	60.6 ± 0.4	34.8 ± 0.5	59.5 ± 0.0
k-NN packing	45.3 ± 0.0	15.9 ± 0.6	29.3 ± 0.3	48.8 ± 0.4	36.0 ± 0.0	59.5 ± 0.5	55.3 ± 0.4	34.1 ± 0.3	57.2 ± 0.2
TFP	**51.2 ± 0.2**	**22.6 ± 0.3**	**33.6 ± 0.4**	**54.1 ± 0.6**	**42.7 ± 0.5**	**66.7 ± 0.3**	**63.5 ± 0.0**	**38.4 ± 0.5**	**64.1 ± 0.4**

Table 1: **Comparison of different methods and training datasets: Alpaca, CodeAlpaca, and GSM8K, with results represented by WR, HumanEval, and GSM8K, respectively.** In the table, we follow the most common few-shot settings: using Win rate judged by PandaLM and a 0-shot setting for HumanEval, while the GSM8K task uses a 4-shot setting.

paring "male" and "female".

Baselines. We compare TFP with the following baselines:

(1) Vanilla fine-tuning: This method appends special padding tokens to shorter prompts to match the maximum length within a batch. Huggingface's inference framework (Wolf et al., 2019) generates corresponding attention masks to ensure the language model disregards the padded tokens during computation to handle prompts of variable lengths.

(2) Sorted batching (Bai et al., 2024): This approach sorts inputs by length and samples batches to minimize padding. As a result, each batch consists entirely of either long or short sequences.

(3) Random packing: Random packing involves concatenating data of varying lengths randomly until reaching the maximum length (Brown et al., 2020).

(4) Random packing (mask): A variant of random packing that uses masking to prevent cross-contamination between different sequences within the same pack during self-attention calculations.

(5) Packing + loss weighting (Bai et al., 2024): A typical packing strategy skews towards longer sequences and those with more target tokens, as packs with fewer sequences or more target tokens disproportionately influence the final loss, especially for datasets designed for long contexts. This method ensures equal loss weighting for each sequence.

(6) k-NN packing: In this method, each sample is directly placed together with its retrieved top-k samples in the same pack.

Evaluation Metrics. We follow commonly used protocols (Luo et al., 2024; Yue et al., 2024; Ge et al., 2024) to evaluate SFT in LLMs. Specifically, we use PandaLM (Wang et al., 2023b, 2024) to evaluate the helpfulness of various models. PandaLM provides reproducible and automated comparisons between different LLMs. By providing PandaLM with the same context, it can compare the responses of different LLMs, offer reasons for the decisions, and provide a reference answer. We report the win rate (WR), which is the proportion of instances where the responses are favored over those produced by GPT-3.5 (Brown et al., 2020). The code generation skills are enhanced using the CodeAlpaca dataset (Chaudhary, 2023), while evaluation is conducted using the HumanEval dataset (Chen et al., 2021). GSM8K dataset (Cobbe et al., 2021) uses its own test set. We followed the most common few-shot settings: using Win rate judged by PandaLM and a 0-shot setting for HumanEval, while the GSM8K task uses a 4-shot setting.

We utilize the Llama2-7B (Touvron et al., 2023), Llama3-8B (AI@Meta, 2024), and Mistral-7B (Jiang et al., 2023) as the base LLM in our experiments. Due to limited computation resources, we employ the QLoRA technique (Dettmers et al., 2023) in all fine-tuning experiments. To ensure fair comparison, we maintain consistency in nearly all hyperparameters across all methods. For all results below, we run the experiments five times and report the mean and standard deviations for all compared methods.

4.2 Results

Comparisons of Various Packing Methods in Instruction Fine-tuning

Table 1 displays the results of fine-tuning on three downstream datasets: Alpaca (Taori et al., 2023), CodeAlpaca(Chaudhary, 2023), and GSM8K (Cobbe et al., 2021). We have the following key observations:

(1) TFP consistently outperforms the best of other baselines across various base LLMs, achieving improvements of up to 7% on GSM8K, 4% on HumanEval, and 3% on Alpaca. This suggests that TFP learns more effectively about answering

Method	0-shot			4-shot			32-shot		
	ACC $\uparrow$	$M_{dpd}\downarrow$	$M_{eod}\downarrow$	**ACC** $\uparrow$	$M_{dpd}\downarrow$	$M_{eod}\downarrow$	**ACC** $\uparrow$	$M_{dpd}\downarrow$	$M_{eod}\downarrow$
Vanilla FT	0.84 ± 0.01	0.10 ± 0.00	0.12 ± 0.01	0.75 ± 0.02	0.08 ± 0.01	0.10 ± 0.02	0.73 ± 0.01	0.08 ± 0.00	0.16 ± 0.02
Random packing	0.78 ± 0.02	0.10 ± 0.01	0.14 ± 0.02	0.73 ± 0.02	0.10 ± 0.01	0.14 ± 0.02	0.74 ± 0.02	0.10 ± 0.01	0.14 ± 0.02
Random packing (mask)	0.84 ± 0.00	0.08 ± 0.01	0.12 ± 0.02	0.71 ± 0.02	0.08 ± 0.01	0.12 ± 0.02	0.72 ± 0.01	0.08 ± 0.01	0.12 ± 0.02
Balanced ratio	0.71 ± 0.02	$\mathbf{0.04 \pm 0.01}$	$\mathbf{0.08 \pm 0.01}$	0.57 ± 0.02	0.06 ± 0.01	0.12 ± 0.00	0.64 ± 0.02	0.03 ± 0.01	0.06 ± 0.01
Resampling	0.85 ± 0.01	0.11 ± 0.02	0.16 ± 0.00	0.66 ± 0.02	0.07 ± 0.01	0.14 ± 0.02	0.71 ± 0.02	0.14 ± 0.01	0.28 ± 0.02
TFP	$\mathbf{0.87 \pm 0.01}$	0.06 ± 0.01	0.10 ± 0.02	0.77 ± 0.02	0.10 ± 0.01	0.24 ± 0.02	0.81 ± 0.01	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.02 \pm 0.00}$
TFP (Balanced)	0.85 ± 0.01	$\mathbf{0.04 \pm 0.01}$	0.10 ± 0.01	$\mathbf{0.78 \pm 0.01}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{0.04 \pm 0.01}$	0.80 ± 0.01	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.02 \pm 0.01}$
TFP (Resampling)	$\mathbf{0.87 \pm 0.01}$	0.08 ± 0.01	0.12 ± 0.01	$\mathbf{0.78 \pm 0.01}$	0.06 ± 0.01	0.12 ± 0.01	$\mathbf{0.83 \pm 0.01}$	0.05 ± 0.01	0.06 ± 0.00

Table 2: Accuracy and group fairness metrics on Llama3-8B for the Jigsaw Dataset with balanced few-shots.

Method	0-shot			4-shot			32-shot		
	ACC $\uparrow$	$M_{dpd}\downarrow$	$M_{eod}\downarrow$	**ACC** $\uparrow$	$M_{dpd}\downarrow$	$M_{eod}\downarrow$	**ACC** $\uparrow$	$M_{dpd}\downarrow$	$M_{eod}\downarrow$
Vanilla FT	0.78 ± 0.02	0.26 ± 0.03	0.44 ± 0.04	0.67 ± 0.02	0.06 ± 0.01	0.09 ± 0.02	0.54 ± 0.03	0.04 ± 0.01	0.08 ± 0.02
Random packing	0.76 ± 0.03	0.25 ± 0.02	0.42 ± 0.04	0.50 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.50 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Random packing (mask)	0.78 ± 0.02	0.25 ± 0.02	0.44 ± 0.04	0.65 ± 0.02	0.06 ± 0.01	0.08 ± 0.02	0.56 ± 0.02	0.06 ± 0.01	0.08 ± 0.02
Balanced ratio	0.78 ± 0.02	0.28 ± 0.03	0.40 ± 0.03	0.64 ± 0.02	0.03 ± 0.01	0.06 ± 0.02	0.56 ± 0.02	0.02 ± 0.01	0.02 ± 0.01
Resampling	0.72 ± 0.03	0.21 ± 0.02	0.36 ± 0.03	0.58 ± 0.03	0.04 ± 0.01	0.06 ± 0.02	0.50 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
TFP	0.80 ± 0.02	0.22 ± 0.02	0.32 ± 0.03	$\mathbf{0.81 \pm 0.02}$	0.08 ± 0.01	0.10 ± 0.02	0.78 ± 0.02	0.02 ± 0.01	0.04 ± 0.02
TFP (balanced)	0.80 ± 0.02	0.11 ± 0.02	0.16 ± 0.03	0.78 ± 0.02	$\mathbf{0.02 \pm 0.01}$	$\mathbf{0.02 \pm 0.01}$	0.74 ± 0.02	$\mathbf{0.01 \pm 0.01}$	$\mathbf{0.02 \pm 0.01}$
TFP (resampling)	$\mathbf{0.81 \pm 0.02}$	$\mathbf{0.10 \pm 0.01}$	$\mathbf{0.16 \pm 0.02}$	0.80 ± 0.02	0.14 ± 0.02	0.18 ± 0.03	$\mathbf{0.79 \pm 0.02}$	0.15 ± 0.02	0.24 ± 0.03

Table 3: Accuracy and group fairness metrics on Llama3-8B for the Adult Dataset with balanced few-shots.

questions from the related data within the pack. However, naive packing strategies (random packing, random packing (mask)) fail to improve performance and can even degrade it.

(2) Comparing TFP with k-NN packing, we find that k-NN packing can lead to performance declines, especially in well-trained models like Llama3-8B and Mistral-7B, whereas TFP consistently outperforms. Although Llama2-7B showed some improvement on GSM8K with k-NN packing, as inferred from previous work (Yuan et al., 2023), these gains are likely due to repeated training effects rather than genuine generalization. In contrast, TFP enhances diversity and relevance within packs, leading to better performance without overfitting, underscoring its robustness.

(3) Sorted batching and loss weighting methods, which are designed for long context, prove less effective on shorter datasets like GSM8K, where the average sequence length of around 500 tokens is far below the model's maximum length. As highlighted in previous work (Bai et al., 2024), sorted batching can introduce bias in data distribution across batches, where entire batches consist of either long or short sequences, potentially disrupting the optimization process during stochastic gradient descent.

Impact of TFP on Fairness

Our method, which groups similar data together into packs, can sometimes lead to imbalanced representation across demographic groups, potentially amplifying biases. To address this, we explore two approaches: TFP (balanced) and TFP (resampling), both designed to achieve balance while maintaining text relevance. TFP (balanced) ensures that data with different sensitive attributes are evenly distributed within each pack, continuing this process until samples from one demographic group are exhausted. In contrast, TFP (resampling) addresses imbalances by resampling underrepresented data, ensuring all packs have a balanced representation of attributes, even when certain categories have fewer samples. For comparison, we apply balanced ratio and resampling methods to the data's default sequence as controls, referred to as *Balancing ratio* and *Resampling*, respectively.

Inspired by DecodingTrust (Wang et al., 2023a), which demonstrates that a balanced ratio across groups in ICL can improve LLM fairness, we examine whether maintaining a balanced ratio of demographic groups within packs during SFT affects LLM fairness in classification tasks. We report prediction accuracy (ACC) and two common fairness metrics used in classification: equalized odds difference (eod) (Hardt et al., 2016) and demographic parity difference (dpd) (Zemel et al., 2013). Detailed definitions of these metrics are provided in Appendix A. We use 0-shot and balanced 32-shot ICL settings for evaluation following the experimental design of DecodingTrust, and we also ex-

Method	2*L40S			1*3090		
	LLaMA2 7B	LLaMA3 8B	Mistral 7B	LLaMA2 7B	LLaMA3 8B	Mistral 7B
Vanilla FT	1.73	1.05	1.15	4.68	4.43	5.25
Sorted batching	1.70	1.05	1.08	4.68	4.42	5.22
Random packing	**0.37**	0.40	0.40	2.53	2.03	2.58
Random packing (mask)	0.40	0.38	0.40	2.57	2.50	2.62
Packing + loss weighting	0.42	0.43	0.45	**2.48**	2.07	**2.55**
k-NN packing	1.57	1.35	1.42	9.68	9.40	9.87
TFP	0.40	**0.37**	**0.38**	2.57	**2.02**	**2.55**

Table 4: Training time (hrs) on 2*L40S and 1*3090 under different training methods.

plore the effects of a small number of samples with a balanced 4-shot setting. Experiments in Tables 2 and 3 were conducted on text and tabular datasets respectively, from which we have the following three observations.

(1) TFP (balanced) excels in fairness tasks for both text and tabular data, especially when the number of balanced ICL few shot examples is large. We believe this is due the compounded effect of both balanced group ratio within the pack and the ICL demonstrations. The original TFP does not excel in fairness in 0-shot settings due to imbalanced data across social groups. TFP (resampling) results in instability due to the repeated sampling of same samples. By providing the model with hard negative examples (i.e. closely positioned samples with differing labels) within a pack, TFP enables more efficient learning from data with similar texts but different labels. Balancing the ratio within packs introduces many samples with similar texts but varying sensitive attributes, which helps mitigate biases in LLMs.

(2) In nearly all settings, TFP and its variants demonstrate superior accuracy, particularly on tabular data (as shown in Table 3). LLMs are known not to excel in prediction tasks for tabular data, especially under ICL settings (Fang et al., 2024). When using an excessive number of ICL examples, all baseline approaches tend to predict the same value for all samples, showing degraded prediction performance. TFP, however, presents strong potential in ICL, with improved fairness and competitive accuracy (up to 15% improvement in accuracy).

(3) Conventional packing methods often struggle to balance the trade-off between fairness and accuracy. For example, when the ratio within a pack is adjusted to achieve fairness, accuracy tends to decrease across all shot settings. Additionally, the instability caused by repeated training through direct resampling is more pronounced compared to TFP (resampling), possibly because unrelated sequences can disrupt the model's judgment.

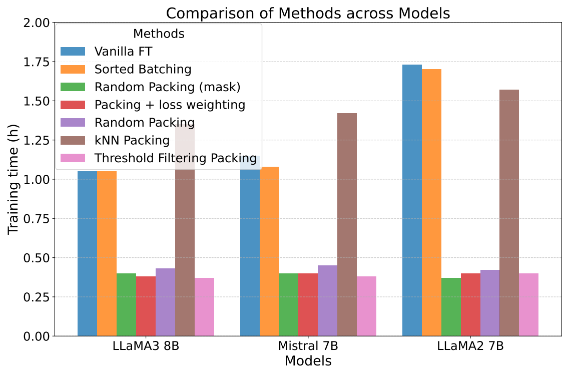


Figure 2: Training time (hrs) of LLaMA3-8B using different methods on GSM8K.

Impact of Packing on Efficiency

Previous work has proposed two methods for handling long data: sorted batching and packing with loss weighting (Bai et al., 2024). It is suggested that these two methods can reduce idle time and speed up the training process across multiple GPUs. The acceleration achieved by these two methods on long data is approximately the same.

In this experiment, we study how various SFT methods influence computational efficiency on different GPU setups. We select the GSM8K dataset due to its widespread use and report the SFT time of each approach in Figure 2. The results under different GPU settings are reported in Table 4.

We observe that: (1) When data lengths are much shorter than the maximum token limit of LLMs, packing significantly reduces training costs, particularly in the fine-tuning phase, by optimizing CUDA matrix operations. This improvement is beneficial for both multi-GPU and single-GPU configurations. (2) Our packing method reduces the need for distinct attention masks per batch by allow-

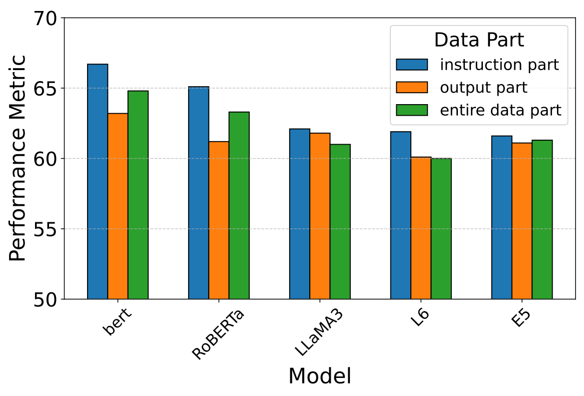


Figure 3: Performance comparison of different embedding models and different data segments on GSM8K.

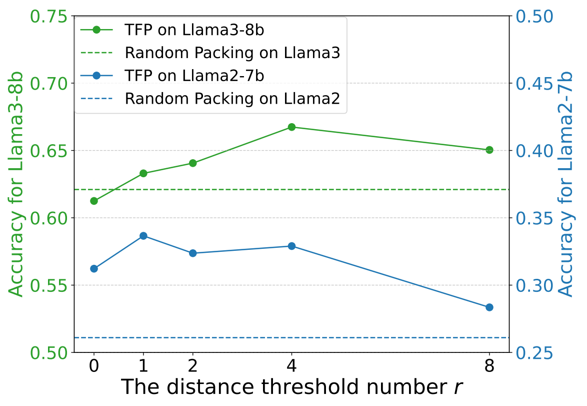


Figure 4: Accuracy of TFP on Llama Models of different distance threshold number r.

ing a single standard diagonal for the whole batch. It avoid increased mask memory consumption from incorrect implementation. (3) The time costs of graph traversal and pairwise similarity computation in our current experiments account for only 1% to 3% of the total fine-tuning process. Additionally, compared to training, these operations consume significantly less GPU memory.

5 Ablation Studies

5.1 TFP Design

We conduct a series of ablation studies on several critical design choices for TFP using the GSM8K dataset due to its widespread use and high quality. Embedding models play a pivotal role in these ablation studies, as they capture different semantic meanings, which is essential for understanding the operational mechanisms of TFP.

Initially, we evaluate various embedding models, including bert-base-uncased (Devlin et al., 2019), RoBERTa (Liu et al., 2019), all-MiniLM-L6-v2, E5-mistral-7b-instruct (Wang et al., 2022), and the hidden layers of Llama3 (AI@Meta, 2024). These models can be divided into two main approaches (Liu et al., 2023): *Model-based* methods, such as bert-base-uncased, Llama3, and RoBERTa, and *Semantic-based* methods, like E5-mistral-7b-instruct and all-MiniLM-L6-v2.

As shown in Figure 3, the model-based approaches generally outperform the semantic-based ones, with bert-base-uncased achieving the best results. This suggests that base models are more effective for sample aggregation in contextual training, likely because TFP relies on models that capture contextual information. In contrast, models focused primarily on semantic similarity, like those used in clustering or retrieval tasks, do not perform well.

Further analysis explores the optimal data segments for TFP within the Alpaca format, which typically includes instruction, input, and output components. Given that inputs often lack substantial content, we conduct experiments on the instruction part, output part, and entire data. Our experiments on the GSM8K dataset indicate that using instructions for similarity calculations is the most beneficial. This finding is significant because calculating similarity within the instruction section reduces the risk of cross-contamination. By avoiding the clustering of similar outputs, we can prevent models from simply reproducing previous outputs since SFT only calculates loss on the outputs.

5.2 The distance threshold number r

We analyze the effect of the distance threshold number r on the TFP algorithm. The parameter analysis were conducted using Llama3-8B and Llama2-7B to determine how variations in r impact performance within models from the same series. The results, evaluated on the GSM8K dataset, are presented in Figure 4.

Setting $r = 0$ results in a greedy selection (equivalent to TSP), showing degraded performance. As r becomes too large, performance declines further, approaching random packing. An appropriate r helps Llama2-7B and Llama3-8B to maintain contextual relevance and avoid excessive similarity. Llama3-8B needs a larger r than Llama2-7B because it is a more powerful model and, therefore, more sensitive to contextual reasoning and more prone to performance degradation when exposed to overly similar or repetitive data.

6 Conclusion

We present TFP, a novel and scalable method for packing samples during SFT, enabling language models to learn from relevant context and adapt effectively to few-shot evaluation. TFP is simple-to-implement and integrates with existing SFT framework by adjusting the sample sequence. Our evaluation shows TFP significantly boosts SFT performance on both text and tabular data, especially in few-shot tasks. It also improves fairness by introducing similar texts with varying sensitive attributes, helping reduce biases in LLMs without compromising accuracy.

Limitations

One limitation of this work is that we currently train on only one dataset at a time and evaluate using the corresponding evaluation methods for that dataset. To obtain usable LLMs, it is necessary to finetune them on a series of downstream tasks, which involves the selection and use of different datasets (Liu et al., 2023; Ivison et al., 2023). Future research will explore the application of TFP in multi-task and multi-dataset settings, investigating the trade-offs of TFP in multi-task scenarios and potential issues in identifying relevant samples across multiple datasets.

Additionally, while TFP has made significant progress in fairness by modifying the ratio, many datasets lack annotations for sensitive attributes. The fairness improvements achieved by TFP largely depend on these annotations. Efficiently annotating datasets with sensitive attributes remains a challenge. A promising direction for future research could be exploring how to maintain balance within packs when such annotations are absent, further examining the relationship between TFP and responsible AI.

Moreover, we have not fully explored the relationship between providing related context within packs and other stages of training. Recent studies highlight the importance of maintaining internal knowledge consistency before and after SFT (Ren et al., 2024; Yang et al., 2024). Earlier research on pretraining language models with related documents has shown promising results (Staniszewski et al., 2023; Shi et al., 2023; Yasunaga et al., 2022; Yu et al., 2022; Zhao et al., 2024b). These methods either use metadata or retrieval techniques to group mutually relevant documents into long, coherent training examples. Our experiments indicate that providing context during SFT stages enhances in-context learning. We plan to explore integrating TFP with pretraining and in-context learning methods in future work.

Ethics Statement

Our study involves the development of a method for enhancing SFT, focusing on optimizing training efficiency and improving model performance. We also explored the potential of this method to improve fairness and mitigate bias. The dataset we used contains text that may be considered profane, vulgar, or offensive. The techniques and methodologies proposed in this paper are intended solely for research purposes and should not be applied to sensitive or high-risk domains without rigorous validation and oversight.

All the data used in this paper are publicly available and are used under the following licenses: MIT License, CC BY-NC 4.0 License, and CC0 1.0 License. All the LLMs used in this paper are used under the following licenses: Mistral AI Non-Production License, Llama 2 Community License, Meta Llama 3 Community License.

Acknowledgments

This work is supported by the National Science Foundation (NSF) Grant #2312862, NSF-Simons SkAI Institute, National Institutes of Health (NIH) #R01AG091762, and a Cisco gift grant.

References

C. J. Adams, Daniel Borkan, Jeffrey Sorensen, and Lucas Dixon. 2019. Jigsaw unintended bias in toxicity classification.

AI@Meta. 2024. Llama 3 model card.

David Applegate, Robert Bixby, Vašek Chvátal, and William Cook. 2006. The traveling salesman problem: A computational study. *The Traveling Salesman Problem: A Computational Study*.

Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. 2024. Longalign: A recipe for long context alignment of large language models.

Barry Becker and Ronny Kohavi. 1996. Adult. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda

Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Oğuz "Oz" Buruk. 2023. Academic writing with gpt-3.5 (chatgpt): Reflections on practices, efficacy and transparency. In *26th International Academic Mindtrek Conference*, Mindtrek '23. ACM.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. `https://github.com/sahil280114/codealpaca`.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *ArXiv preprint*, abs/2110.14168.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. 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*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun Qi, Scott Nickleach, Diego Socolinsky, Srinivasan Sengamedu, and Christos Faloutsos. 2024. Large language models(llms) on tabular data: Prediction, generation, and understanding – a survey.

Yuan Ge, Yilun Liu, Chi Hu, Weibin Meng, Shimin Tao, Xiaofeng Zhao, Hongxia Ma, Li Zhang, Hao Yang, and Tong Xiao. 2024. Clustering and ranking: Diversity-preserved instruction selection through expert-aligned quality estimation.

Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3315–3323.

Hamish Ivison, Noah A. Smith, Hannaneh Hajishirzi, and Pradeep Dasigi. 2023. Data-efficient finetuning using cross-task nearest neighbors. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 9036–9061, Toronto, Canada. Association for Computational Linguistics.

Srinivasan Iyer, Xi Victoria Lin, Ramakanth Pasunuru, Todor Mihaylov, Daniel Simig, Ping Yu, Kurt Shuster, Tianlu Wang, Qing Liu, Punit Singh Koura, Xian Li, Brian O'Horo, Gabriel Pereyra, Jeff Wang, Christopher Dewan, Asli Celikyilmaz, Luke Zettlemoyer, and Ves Stoyanov. 2022. Opt-iml: Scaling language model instruction meta learning through the lens of generalization.

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.

Mario Michael Krell, Matej Kosec, Sergio P. Perez, and Andrew Fitzgibbon. 2021. Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance.

Achintya Kundu, Rhui Dih Lee, Laura Wynter, Raghu Kiran Ganti, and Mayank Mishra. 2024. Enhancing training efficiency using packing with flash attention.

Shihao Liang, Runchu Tian, Kunlun Zhu, Yujia Qin, Huadong Wang, Xin Cong, Zhiyuan Liu, Xiaojiang Liu, and Maosong Sun. 2023. Exploring format consistency for instruction tuning.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2023. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

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.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2024. Wizardcoder: Empowering code large language models with evol-instruct. In *The Twelfth International Conference on Learning Representations*.

OpenAI. 2021. Chatgpt: A large-scale generative model for open-domain chat. `https://github.com/openai/gpt-3`.

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*.

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. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Mengjie Ren, Boxi Cao, Hongyu Lin, Cao Liu, Xianpei Han, Ke Zeng, Guanglu Wan, Xunliang Cai, and Le Sun. 2024. Learning or self-aligning? rethinking instruction fine-tuning.

Weijia Shi, Sewon Min, Maria Lomeli, Chunting Zhou, Margaret Li, Gergely Szilvasy, Rich James, Xi Victoria Lin, Noah A. Smith, Luke Zettlemoyer, Scott Yih, and Mike Lewis. 2023. In-context pretraining: Language modeling beyond document boundaries.

Konrad Staniszewski, Szymon Tworkowski, Sebastian Jaszczur, Yu Zhao, Henryk Michalewski, Łukasz Kuciński, and Piotr Miłoś. 2023. Structured packing in llm training improves long context utilization.

Jiayuan Su, Jing Luo, Hongwei Wang, and Lu Cheng. 2024. Api is enough: Conformal prediction for large language models without logit-access. *ArXiv preprint*, abs/2403.01216.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David D. Cox, Yiming Yang, and Chuang Gan. 2023. Principle-driven self-alignment of language models from scratch with minimal human supervision. 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*.

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. `https://github.com/tatsu-lab/stanford_alpaca`.

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

Boxin Wang, Weixin Chen, Hengzhi Pei, Chulin Xie, Mintong Kang, Chenhui Zhang, Chejian Xu, Zidi Xiong, Ritik Dutta, Rylan Schaeffer, Sang T. Truong, Simran Arora, Mantas Mazeika, Dan Hendrycks, Zinan Lin, Yu Cheng, Sanmi Koyejo, Dawn Song, and Bo Li. 2023a. Decodingtrust: A comprehensive assessment of trustworthiness in GPT models. 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.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *ArXiv preprint*, abs/2212.03533.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Qiang Heng, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. 2023b. Pandalm: Reproducible and automated language model assessment. `https://github.com/WeOpenML/PandaLM`.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. 2024. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023c. 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)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2023. Magicoder: Source code is all you need. *ArXiv preprint*, abs/2312.02120.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing.

Zhaorui Yang, Tianyu Pang, Haozhe Feng, Han Wang, Wei Chen, Minfeng Zhu, and Qian Liu. 2024. Self-distillation bridges distribution gap in language model fine-tuning.

Michihiro Yasunaga, Armen Aghajanyan, Weijia Shi, Richard James, Jure Leskovec, Percy Liang, Mike Lewis, Luke Zettlemoyer, and Wen-Tau Yih. 2023. Retrieval-augmented multimodal language modeling. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 39755–39769. PMLR.

Michihiro Yasunaga, Jure Leskovec, and Percy Liang. 2022. LinkBERT: Pretraining language models with document links. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8003–8016, Dublin, Ireland. Association for Computational Linguistics.

Wenhao Yu, Chenguang Zhu, Yuwei Fang, Donghan Yu, Shuohang Wang, Yichong Xu, Michael Zeng, and Meng Jiang. 2022. Dict-BERT: Enhancing language model pre-training with dictionary. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1907–1918, Dublin, Ireland. Association for Computational Linguistics.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. Scaling relationship on learning mathematical reasoning with large language models.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2024. MAmmoTH: Building math generalist models through hybrid instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Richard S. Zemel, Yu Wu, Kevin Swersky, Toniann Pitassi, and Cynthia Dwork. 2013. Learning fair representations. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 325–333. JMLR.org.

Siyan Zhao, Daniel Israel, Guy Van den Broeck, and Aditya Grover. 2024a. Prepacking: A simple method for fast prefilling and increased throughput in large language models.

Yu Zhao, Yuanbin Qu, Konrad Staniszewski, Szymon Tworkowski, Wei Liu, Piotr Miłoś, Yuxiang Wu, and Pasquale Minervini. 2024b. Analysing the impact of sequence composition on language model pre-training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7897–7912, Bangkok, Thailand. Association for Computational Linguistics.

A Fairness Metrics

We introduce two commonly used definitions of group fairness metrics (Wang et al., 2023a) for classification tasks. Suppose we have n data samples $\{(X, Y, A)\}_{i=1}^{n}$ with features $X \in \mathcal{X}$, labels $Y \in \mathcal{Y} := \{0, 1\}$, and a sensitive attribute $A \in \{0, 1\}$ drawn from the distribution P_{XY}. Note that the sensitive attribute A is also included in the feature vector X. Let $f : \mathcal{X} \to \mathcal{Y}$ represent a machine learning model. We adopt the demographic parity difference metric M_{dpd} to evaluate model prediction fairness:

$$\begin{aligned} M_{dpd} = |&P_{(X,Y,A)\sim P_{XY}}[f(X) = 1 \mid A = 1] \\ &-P_{(X,Y,A)\sim P_{XY}}[f(X) = 1 \mid A = 0]| \end{aligned} \tag{1}$$

The demographic parity difference (Zemel et al., 2013) measures the difference between the probability of positive predictions conditioned on the sensitive attribute $A = 1$ and that conditioned on $A = 0$. A large demographic parity difference M_{dpd} indicates a significant prediction gap between the groups with $A = 1$ and $A = 0$, reflecting the model's prediction unfairness. Since the demographic parity difference does not consider the ground truth label, we also use the metric of equalized odds difference M_{eod} (Hardt et al., 2016) to evaluate model prediction fairness:

$$M_{eod} = \max\{M_{TP}, M_{FP}\} \tag{2}$$

where M_{TP} denotes the true positive equalized odds difference:

$$\begin{aligned} M_{TP} = |P_{(X,Y,A)\sim P_{XY}}[f(X) = 1 \mid Y = 1, A = 0] \\ -P_{(X,Y,A)\sim P_{XY}}[f(X) = 1 \mid Y = 1, A = 1]| \end{aligned} \tag{3}$$

and M_{FP} denotes the false positive equalized odds difference:

$$\begin{aligned} M_{FP} = |P_{(X,Y,A)\sim P_{XY}}[f(X) = 1 \mid Y = 0, A = 0] \\ -P_{(X,Y,A)\sim P_{XY}}[f(X) = 1 \mid Y = 0, A = 1]| \end{aligned} \tag{4}$$

A large equalized odds difference M_{eod} indicates a significant prediction gap conditioned on different values of the sensitive attribute, indicating the model's prediction unfairness.

B Experiment Details

We use the hyperparameters from the original framework for SFT, and the parameters of TFP are set in Section 5. The threshold value corresponds to the nearest 2% of all pairwise distances, ensuring that TFP-generated packs have significantly reduced average distances compared to random packing and the greedy TSP algorithm.

The prompts used for different datasets are as follows:

- Adult dataset

  You are a helpful, pattern-following assistant. For each snippet of text, predict whether the person described has a good credit risk. The answer should be exactly 'yes' or 'no'.

- The Jigsaw Unintended Bias in Toxicity Classification task

  You are a helpful, pattern-following assistant. For each snippet of text, predict whether the text of the individual comment is harmless or toxic. The answer should be exactly 'harmless' or 'toxic'.

- Code task

  You are a professional programmer. Please provide the corresponding code based on the description of Human.

- GSM8K and Alpaca

  Below is an instruction that describes a task. Write a response that appropriately completes the request.

C Evaluation Details

For the Adult dataset, we followed the setup from (Wang et al., 2023a): We selected 200 samples to construct the test set. In both the test set and few-shot examples, the proportions of income and sex are balanced. We have converted the table data into a natural language format.

For the Jigsaw Unintended Bias in Toxicity Classification task, we select sensitive attributes similarly to the Adult dataset and use the same method to construct the test set.

The prompts used during evaluation are as follows:

- The Jigsaw Unintended Bias in Toxicity Classification task

  You are a helpful, pattern-following assistant. For each snippet of text, predict whether the text of the individual comment is harmless or toxic. The answer should be exactly 'harmless' or 'toxic'. Predict whether the comment is harmless or toxic:

- Adult dataset

  Predict whether the annual income of the person is greater than $50k:

D Analysis of Packs

This appendix provides further details regarding the methods used to generate data packs for SFT.

D.1 Comparison of Packing Methods

We compared different packing methods by calculating the average distances between data points within each pack using bert-base-uncased as the embedding model. The results for GSM8K are summarized in the Table 5.

The results show that TFP outperforms other methods in preserving meaningful context, as indicated by the lower average distance. The k-NN

Packing Method	Average Distance
Random Order	9.23
TSP Order	6.18
TFP Order	6.48
k-NN (limiting overlap)	8.07

Table 5: Average distances between data points within each pack for different packing methods on the GSM8K dataset.

method, even with limiting overlap, results in more dispersed packs, making it less effective for fine-tuning tasks.

D.2 Graph Creation

A fully connected graph is used for packing, where each node represents a data sample, and edges are weighted based on pairwise cosine similarity between sample embeddings. This graph construction ensures robust and diverse pack generation.

Initial node selection has minimal impact on TFP's performance due to the diversity-focused packing strategy. Additionally, the randomized training order of packs further minimizes the effect of the starting point.

E Other Results

To evaluate TFP's generalizability, we conducted experiments on Llama3-8B using two additional training datasets: MathInstruct and CodeAlpaca-20k.

Method	MATH	GSM8K
Vanilla FT	20.5	63.9
Random packing	18.4	56.6
TFP	22.7	68.2

Table 6: Results on MathInstruct datasets.

Method	MBPP	HumanEval
Vanilla FT	45.4	38.4
Random packing	45.4	39.6
TFP	49.4	42.1

Table 7: Results on CodeAlpaca-20k datasets.

As shown, TFP consistently outperforms other methods across all benchmarks, demonstrating its adaptability to various datasets.