Textual Dataset Distillation via Language Model Embedding

Yefan Tao Chris Kong Andrey Kan Laurent Callot
{tayefan, luyankon, avkan, lcallot}@amazon.com
Amazon Web Services, Seattle, Washington, USA

Abstract

Dataset distillation is a process aimed at condensing datasets while preserving essential characteristics. In the text domain, prevailing methods typically generate distilled data as embedding vectors, which are not human-readable. This approach simplifies optimization but limits the transferability of distilled data across different model architectures. To address this limitation, we introduce a model-agnostic, data-efficient method that leverages Language Model (LM) embeddings. Compared to parameter-efficient methods such as LORA, our approach achieves comparable performance with significantly faster processing times. We evaluate our methodology through classification tasks on datasets like IMDB and AG-News, demonstrating performance that is on par with or exceeds previous model-dependent techniques. By utilizing LM embeddings, our method offers enhanced flexibility and improved transferability, expanding the range of potential applications.

1 Introduction

Recent advancements in Natural Language Processing (NLP) models have been significantly driven by the availability of large-scale datasets(Wu et al., 2020) (Yin et al., 2021). It has been widely observed that larger datasets often lead to better performance. However, there is a growing recognition that "small and high quality data" can be more effective than simply "larger but low-quality data"(Gunasekar et al., 2023; Li et al., 2023). Additionally, training NLP models on large-scale datasets can be remarkably time-consuming and resource-intensive, presenting a substantial challenge in the field. This has spurred ongoing research efforts to optimize the efficiency of model training while ensuring the data quality and integrity.

In the context of recent trends involving Large Language Models (LLMs), the consideration of dataset size and quality becomes particularly pertinent. LLMs, despite their extensive capabilities, are associated with inherent challenges, including significant latency in API calls and substantial operational costs. Thus it has been increasingly common to use a LLM as an orchestrator that breaks

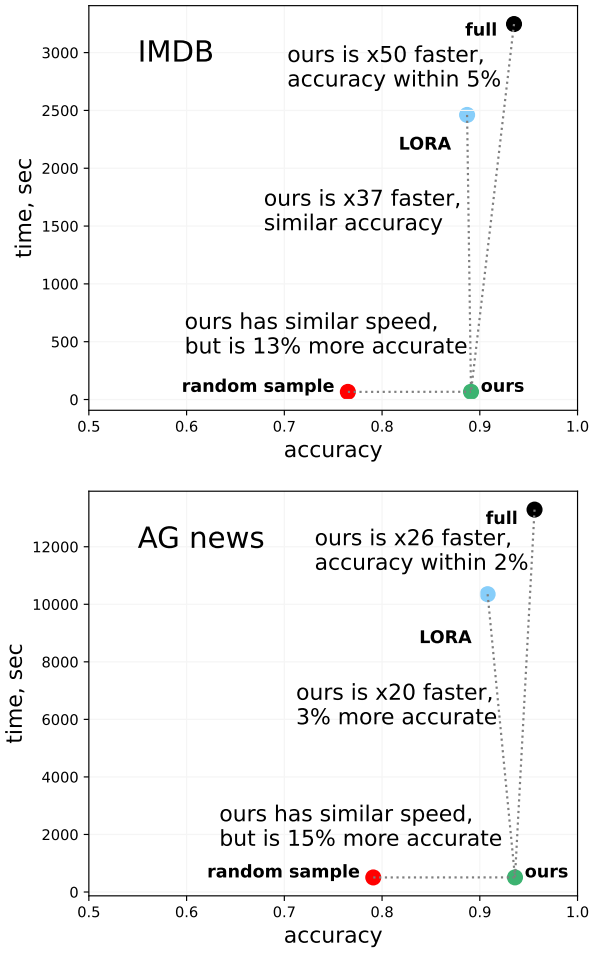

Figure 1: Comparative training time of four training strategies on IMDB and AG-News datasets. Our proposed method (trained with a small distilled dataset) achieves significantly faster training times than full data and LORA, with comparable accuracy, and outperforms training with a random subset in accuracy at similar speeds. Detail in Section 5.7

Findings of the Association for Computational Linguistics: EACL 2024, pages 12557–12569

down natural language request into steps, and invokes relatively small, task-specific models for individual sub-tasks. For example, in multifunctional enterprise chatbots, it is common to first identify query intent using a fine-tuned intent classifier (e.g., the entered query can be classified into "taking action on data", "answering a question about data", or "off topic"). Using a specialized model is cheaper and faster compared to using LLM for the same task. Moreover a model fine-tuned on customer data can be more effective on customer's domain compared to a large generic model. Such a synergy between pre-trained LLMs and specialized fine-tuned models enhances overall system efficacy. In turn, small, high-quality datasets are crucial for training highly specialized and efficient models(Li et al., 2023).

One promising approach to obtain "small and high quality data" is dataset distillation(Wang et al., 2020; Li et al., 2022), a technique that aims to distill the knowledge from a large dataset into a smaller, more compact dataset that preserves the performance of the model. Unlike traditional core-set selection method, dataset distillation aims at generating "synthetic data" which represents the whole dataset's information. The main benefits of dataset distillation comes from two aspects: 1) remove the requirement of heuristics, which is always needed in core-set selection algorithm 2) not bounded by real data points, which leads to higher compression rate.

While LoRA (Low-Rank Adaptation)(Hu et al., 2021) primarily focuses on the parameter aspect by reducing the number of trainable parameters to lower memory and computational demands, dataset distillation targets data efficiency. Unlike LoRA, which modifies the model architecture to enhance parameter efficiency, dataset distillation reduces the volume of training data, thereby significantly enhancing training speed (as shown in Figure 1), which is particularly valuable in scenarios requiring rapid deployment or frequent retraining. This distinction underscores the unique benefits each method offers in optimizing model training under different operational constraints.

Most existing dataset distillation methods have been predominantly developed for the vision domain. A significant challenge in distilling textual data arises from the discrete nature of real text, which prevents direct training on synthetic data. To circumvent this problem, previous research (Sucholutsky and Schonlau, 2021; Li and Li, 2021) has focused on generating 'synthetic embeddings' as a form of distilled data. While this approach yields reasonable results, it comes with a severe limitation: these synthetic embeddings are typically constrained to the same model architecture and tokenizer used during their generation. This restriction greatly impedes the practical applications of such methods. Ideally, distilled textual data should be universally effective, robust against variations in model architecture, and adaptable across different NLP models. Such universality is crucial for advancing realistic applications in textual dataset distillation.

Here we focus on the question: "how to perform an effective model-independent distillation of text data?" To this end, we are proposing a novel three-step procedure. Initially, the original textual dataset is encoded into vector representations using state-of-the-art embedding models (e.g., Sentence-BERT, OpenAI embedding, etc.). Next, the data is strategically sampled, aiming to maximize information retention. This step is not limited to selecting subsets of existing vectors: it permits an unconstrained search within the embedding space. Finally, the distilled embedding vectors are translated back into authentic textual format through a "vec2text" model. The regenerated text is crafted to retain the integrity of the information for application across various downstream tasks. We will go through each step in detail in Section 3. We named the proposed method DaLLME, which stands for **Da**ta Distillation via **L**arge **L**anguage **M**odel **E**mbedding.

Our primary contributions are as follows:

- We introduce a novel model-agnostic framework that leverages the embeddings from Large Language Models (LLMs) to distill comprehensive information from extensive textual datasets into a more compact, synthetic form.
- We demonstrate that the application of 'prompting' techniques can enhance the quality of embedding representations for specialized downstream tasks.
- Through rigorous evaluation across a variety of downstream tasks, our approach achieves state-of-the-art performance on multiple textual datasets, underscoring the efficacy of our proposed method.

2 Related Works

2.1 Dataset Distillation

Dataset distillation has been a topic of increasing interest in recent years, as the size and complexity of machine learning datasets continue to grow. Most of the previous work in dataset distillation requires the training on synthetic data to mimic the behavior of the training on real data. The first line of work (Wang et al., 2020) treat the data and learning rate as part of the trainable parameters, then the synthetic data is generate by minimizing the loss with second-order derivative. (Sucholutsky and Schonlau, 2021) extends the work to include label as "soft-label". Also, (Sucholutsky and Schonlau, 2021) is the first work to apply the dataset distillation algorithm in the text domain. However, these works both requires bi-level iteration and second-order derivatives calculation, which is very time-consuming and not scalable.

To generate synthetic data efficiently, many following works avoid the second-order iteration and propose novel approaches (Lei and Tao, 2023). One line of work is proposing new meta-learning frameworks: (Bohdal et al., 2020) and (Sucholutsky and Schonlau, 2021) learns the label instead of data to avoid the heavy weight updates. (Nguyen et al., 2021) and (Nguyen et al., 2022) use the kernel methods in dataset distillation. The other line of work is focusing on data matching framework, including gradient matching (Zhao et al., 2021), (Lee et al., 2022), distribution matching(Zhao and Bilen, 2022; Wang et al., 2022), trajectory matching(Cazenavette et al., 2022), (Du et al., 2023), etc.

2.2 Core-Set selection

Core-Set selection methods aim to identify and extract a representative subset of the full dataset, known as a 'Core-Set,' which succinctly captures the essence of the entire dataset while substantially reducing its size(Feldman, 2020). The core idea is to maintain the distribution and diversity of the original data, allowing for the preservation of learning complexity and ensuring minimal loss in model performance when trained on this reduced set. Core-Set selection methods can be categorized based on their sampling criteria, including geometry-based(Sener and Savarese, 2017; Killamsetty et al., 2021b), uncertainty-based(Coleman et al., 2019), submodularity-based(Mirrokni and Zadimoghaddam, 2015), gradient-based(Aljundi et al., 2019; Killamsetty et al., 2021a) and loss approximation-based(Guo et al., 2022).

2.3 Vec2Text

"Vec2text" refers to the process of converting numerical vector representations into human-readable text. The Encoder-Decoder architecture has been highly successfully in the image processing domain, exemplified by its ability to encode images into high–dimensional hidden vectors and subsequently decode them back to image form(Badrinarayanan et al., 2017). Within the realm of Natural Language Processing(NLP), similar Encoder-Decoder frameworks are prevalent and are foundational to tasks such as machine translation(Cho et al., 2014) and text summarization(Liu and Lapata, 2019). However, the landscape of NLP is evolving with Large Language Models (LLMs) increasingly adopting either Encoder-only(like BERT and RoBERTa) or Decoder-only (GPT-family, LLaMA) architectures to fulfill diverse objectives.

A body of research has dedicated efforts to construct vec2text models tailored to specialized datasets. One approach involves the development of these models via auto-encoders(Bowman et al., 2015; Shen et al., 2020), while another line of inquiry seeks to establish a universal vec2text model by training on extensive text corpora(Cideron et al., 2022). More recently, a novel methodology has emerged wherein a multi-step correction strategy is employed to refine and approximate the translation of embedding vectors back into coherent plain text (Morris et al., 2023).

3 Methodology

3.1 Problem Statement

Given a large-scale dataset $D_o = \{x_i\}^N$, where N is the number of sample in D_o, the problem of dataset distillation is to generate a synthetic data $D_s = \{\tilde{x_j}\}^M$ with $M \ll N$, such that D_s effectively encapsulates the informational essence of D_o. The distilled dataset D_s should enable the training of models that achieve comparable performance to models trained on D_o, despite the significant reduction in size. Both x_i and $\tilde{x_j}$ are human-readable text so that the generated synthetic data can be transferred to different downstream models. Formally,

$$D_s = \arg\min_{D_s} eval(\Theta(D_o)) - eval(\Theta(D_s)) \quad (1)$$

where $\Theta(D_o)$ and $\Theta(D_o)$ are models trained on

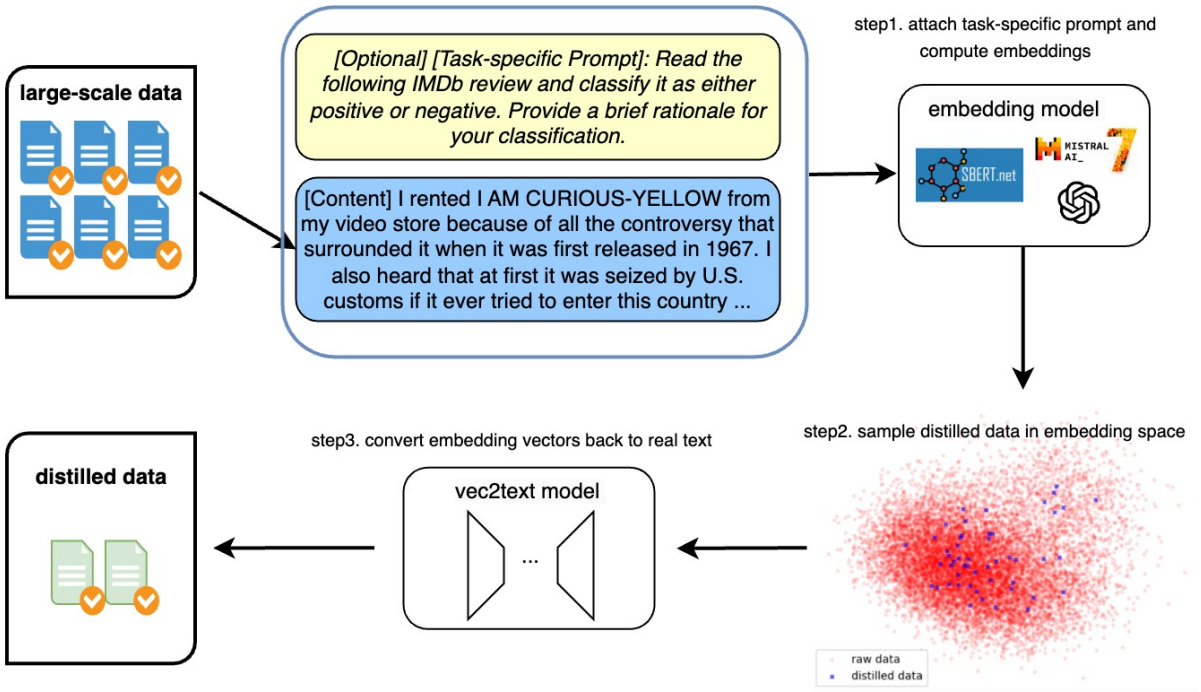

Figure 2: The DaLLME Framework Process Flow. The framework initiates by transforming raw textual data into embedding vectors using a Language Model. Subsequently, a condensed set of distilled vectors is derived in the embedding space, designed to encapsulate maximum informational content. Finally, the 'vec2text' model translates these distilled embedding vectors back into textual form. The resulting distilled text is not only significantly more compact but also retains the informational richness of the original, expansive dataset, making it suitable for various downstream tasks.

D_o and D_s for specific downstream tasks, $eval$ is the evaluation function for downstream task.

3.2 Proposed Method

We propose an innovative approach to address this challenge by utilizing Language Model embeddings. These embeddings are known for encapsulating rich information, as highlighted by (Morris et al., 2023). Furthermore, they facilitate quantitative operations, enabling advanced data manipulation. Our methodology comprises three key steps:

Text-to-Embedding Conversion: In the first step, we convert textual data into embedding vectors. This transformation exploits the Large Language Model's (LLM) proficiency in distilling and representing intricate textual content within a multidimensional vector space. It is well-established that judicious prompting can substantially enhance the LLM's efficacy in downstream tasks(Kojima et al., 2022). We posit that embeddings, when enriched with appropriate prompts, encapsulate a richer informational content compared to their non-prompted counterparts.

For scenarios where dataset distillation is tailored for a specific downstream task, we incorporate a task-specific prompt to augment the performance of the embedding model. This strategic prompting is designed to align closely with the task's nuances, thereby boosting the model's relevance and effectiveness. Conversely, in the context of general-purpose dataset distillation, the addition of prompts is deemed unnecessary. In such cases, our focus shifts to maintaining the versatility and broad applicability of the embeddings without the influence of task-specific cues.

Synthetic Embedding Generation: In the second step, we generate synthetic data within the embedding space. The overarching goal of this step is a general yet vital one: given a list of high-dimensional vectors, we aim to generate a compact set that captures the essence and information contained in the original vector list. This process is akin to distilling the key features and patterns from a vast dataset into a more manageable and insightful subset.

To support this approach, we introduce a Generator, represented as g, which is designed to be adaptable to specific tasks. A good generation strategy should offer users the flexibility to control the generation size, allowing for adjustments based on

specific needs and constraints. In our approach, we utilize the k-centroid algorithm, which does not require training (detail in Appendix B.2). We also experimented with the Gaussian Mixture Model (GMM) (Bond et al., 2001) and observed no significant difference in results.

The choice of the generator, whether it be k-centroid, or any other method, can be tailored to the specific requirements of the task at hand. This adaptability ensures that our approach is not only versatile but also capable of yielding high-quality synthetic data that is both relevant and representative of the original dataset.

Embedding-to-Text Translation: In the last step, we convert the synthetic embedding vectors back into human-readable text, which ensures its generalization across different model architectures. We adopted the "multi-step correction" strategy proposed by (Morris et al., 2023), and trained several vec2text models for specific down-stream tasks. This step is crucial for the transferability of the distilled data.

4 Experiments Configuration

Embedding Model. MTEB(Muennighoff et al., 2022) provides a comprehensive benchmarking on a series of embedding models' performance on different NLP tasks. Generally, the dimension of embedding model correlates to the model's representative ability. In order to understand the embedding model's performance on the quality of distilled dataset, we picked the following embedding models with different output dimensions: **GloVe**[1](dim=300), **e5-base-v2**[2](dim=768) from Sentence-Transformer, **text-embedding-ada-2-002**(dim=1536) and **text-embedding-3-large**[3](dim=3072) from OpenAI API.

Training of vec2text Model. The Vec2Text model functions primarily as a decoder and requires training across various embedding models. Following (Morris et al., 2023), we initialize the model from T5-Base and fine-tune with domain-specific training data. We set the maximum sequence length as 128 to avoid model deterioration. To ensure the vec2text model's efficacy, we employ the BLEU score to access similarity between the original and the reconstructed text. Each vec2text model is fully trained until the BLEU score is converged on testing data.

Downstream Tasks and Datasets. We evaluate the proposed method with the text classification task, focusing on the IMDB[4] and AG-News[5] datasets. IMDB is a dataset for binary sentiment classification, featuring 50,000 movie reviews labeled as positive or negative. AG-News comprises news articles across four categories, aiming for topic classification.

Evaluation. Our goal is to synthesize a dataset that maintains its effectiveness across a diverse downstream models, thereby ensuring robustness and model-agnostic performance. To this end, we have conducted evaluations using different downstream models, including Logistic Regression (LR), Naive Bayes (NB), Support Vector Machine (SVM), more advanced neural network models including customized TextCNN, TextRNN and pre-trained language models, including distilBERT(Sanh et al., 2019) and T5-Base(Raffel et al., 2020). For input text vectorization, LR/NB/SVM utilize TF-IDF, whereas TextCNN/TextRNN employ GloVe-100d embeddings, and distilBERT/T5-Base leverage their own tokenizer and embeddings. Utilizing the distilled dataset, we trained these downstream models to full capacity and assessed their performance by measuring accuracy on the original test dataset. Higher accuracy indicates that the distilled data retains a high level of informational content. For each result, we conducted 10 experiments and reported the average accuracy along with its standard deviation. More detail about the model architectures are in Appendix A.

Baseline. There are limited works in the domain of textual dataset distillation, and previous work(Sucholutsky and Schonlau, 2021; Li and Li, 2021) are all model-specific. Even though it's not a fair comparison, we still include it to better understand whether the proposed method can compete with the model-specific ones. Meanwhile, we also add random sampling as a naive baseline. Overall, we have the following baselines:

- Random Sampling: We randomly select a subset from the original dataset, repeat the experiment 10 times, and average the results to reduce sampling error.

[1]https://huggingface.co/sentence-transformers/average_word_embeddings_glove.6B.300d

[2]https://huggingface.co/intfloat/e5-base-v2

[3]https://platform.openai.com/docs/guides/embeddings

[4]https://huggingface.co/datasets/imdb

[5]https://huggingface.co/datasets/ag_news

		method	LR	SVM	Naive-Bayes	TextCNN	TextRNN	distilBERT	T5-Base	average
IMDB		Full Data	0.883(00)	0.874(00)	0.840(00)	0.871(18)	0.833(21)	0.907(15)	0.935(17)	0.877(14)
	baseline	Random	0.583(19)	0.578(21)	0.585(29)	0.557(20)	0.531(23)	0.608(27)	0.624(26)	0.581(24)
		TDD	-	-	-	**0.739**	-	-	-	-
		EDA	0.591(20)	0.591(21)	0.627(19)	0.625(19)	0.585(23)	0.645(23)	0.743(19)	0.630(20)
		DeepCore	0.593(20)	0.595(20)	0.585(33)	0.569(21)	0.539(17)	0.734(23)	0.706(15)	0.617(22)
	proposed	GloVE-300d	0.640(21)	0.636(22)	0.636(28)	0.608(18)	0.573(23)	0.675(20)	0.651(21)	0.631(22)
		e5-base	0.605(20)	0.624(18)	0.620(22)	0.599(19)	0.585(20)	0.685(23)	0.697(19)	0.631(20)
		openAI-ada-002	**0.651(19)**	0.670(17)	0.653(24)	0.698(22)	**0.657(21)**	0.758(18)	0.762(17)	0.692(20)
		openAI-3-large	0.650(18)	**0.677(19)**	**0.684(21)**	0.683(19)	0.645(24)	**0.769(16)**	**0.763(22)**	**0.696(20)**
AG-News		Full Data	0.905(01)	0.907(00)	0.889(00)	0.913(15)	0.921(17)	0.945(13)	0.954(15)	0.919(11)
	baseline	Random	0.677(10)	0.678(10)	0.686(07)	0.752(10)	0.741(14)	0.786(17)	0.763(15)	0.726(12)
		TDD	-	-	-	0.856	-	-	-	-
		EDA	0.680(10)	0.679(10)	0.700(13)	0.853(06)	0.814(07)	0.837(09)	0.830(12)	0.770(10)
		DeepCore	0.671(10)	0.671(11)	0.681(12)	0.811(07)	0.783(17)	0.831(22)	0.811(19)	0.751(15)
	proposed	GloVE-300d	0.685(17)	0.695(15)	0.689(23)	0.857(18)	0.813(16)	0.854(19)	0.847(21)	0.777(19)
		e5-base	0.703(15)	0.719(21)	0.723(18)	0.860(19)	0.821(18)	0.866(19)	0.861(23)	0.793(19)
		openAI-ada-002	**0.750(19)**	0.750(21)	0.757(18)	0.874(21)	**0.847(23)**	0.878(18)	0.877(18)	0.819(20)
		openAI-3-large	0.746(19)	**0.755(18)**	**0.764(23)**	**0.883(18)**	0.845(19)	**0.889(16)**	**0.893(20)**	**0.825(19)**

Table 1: Accuracy comparison of a Distilled Dataset at 0.1% of the Original Size(20 for IMDB and 120 for AG-News) in Text Classification Tasks. EDA uses 10 times the 0.1% size after augmentation. Results of TDD is from (Sucholutsky and Schonlau, 2021) and (Li and Li, 2021). 0.877(14) means 0.877±0.014

- Text Dataset Distillation(TDD) (Sucholutsky and Schonlau, 2021; Li and Li, 2021): The "traditional" dataset distillation method performs gradient descent on model-specific synthetic data, generating a numerical matrix instead of plain text.

- Easy Data Augmentation (EDA) (Wei and Zou, 2019): We randomly sample a subset from the full dataset and augment it tenfold using simple text editing techniques, including synonym replacement, random insertion, random swap, and random deletion.

- DeepCore (Guo et al., 2022): A state-of-the-art core-set selection method, originally proposed for image data, is adapted here for textual data by treating it as a one-dimensional 'image' and applying the 'Submodularity-based' method.

Other implementation details are in Appendix B.1.

5 Results and Discussion

Main Results. We generated distilled dataset with a size of 0.1% of the full data, and evaluated the accuracy with different down-streamed models. The accuracy attained using full datasets establishes the upper performance limit for each architecture. `text-embedding-3-large` achieve the highest averaged accuracy. Compared to the accuracy on full data, we recover the accuracy by 79.2% on IMDB and 89.9% on AG-News. While the random sampling only recovers 64.6% on IMDB and 78.9% AG-News. Compared to the model-specific method TDD, our method is worse on IMDB by 4.2% but better on AG-News by 2.9%.

5.1 Comparison between data augmentation and data distillation

From Table 1, our proposed method outperforms EDA under all configurations. Both methods aim to improve model accuracy with limited training data, but they approach this goal from different directions. Data augmentation increases the training data size to achieve performance gains or robustness. On the other hand, dataset distillation reduces the training data size, making training more efficient in terms of time and cost, even with an acceptable performance drop. Augmentation is particularly useful when there is indeed limited training data available, as it enhances the dataset by generating more samples. Conversely, distillation is more beneficial when the goal is to reduce the training size from a large dataset, as it condenses information from the entire dataset. Under the same constraint of training data, distillation is superior to augmentation because it effectively captures and compresses the essential information, whereas augmentation only utilizes information from a small selected subset. This is similar to core-set selection, which also uses information from a selected subset, albeit a more informative one than random sampling.

5.2 Effect of Distillation Ratio on recovered accuracy

We define the **Distillation Ratio** as the ratio between the size of distilled dataset and the size of full dataset. We evaluated the recovered accuracy at different distillation ratio. Figure 4 and Figure 5 in Appendix C show the results on IMDB and AG-News respectively. At all distillation ratios, the proposed method achieves better accuracy under different distillation ratio, which confirms the dense information in the distilled data. Specifically, with a 10% dataset size on IMDB, we can recover up to 98.5% of the average accuracy achieved using the full dataset. Similarly, with a 1% dataset size on AG-News, we can recover up to 95.4% of the average accuracy.

5.3 Effect of embedding model

The quality of embedding model is crucial to the distillation process. In Figure 4 and Figure 5, we see that `text-embedding-ada-002` and `text-embedding-3-large` consistently outperform `GloVE` and `e5-base`. Our assumption is that these models have a higher dimension, which means they have stronger representative ability. To verify that, we train a simple linear classifier, which takes the embedding vectors as input and predicts the classification label. Then we report the resulting accuracy from different embedding models. Higher accuracy means stronger capability in the classification task. Results are reported in Table 2. We can see that the embedding model's quality improve in the order of `GloVE.6B.300d` $\rightarrow$ `e5-base-v2` $\rightarrow$ `text-embedding-ada-002` $\rightarrow$ `text-embedding-3-large`. This confirms our assumption and we conclude that **better embedding model leads to better distilled dataset**.

Model	Acc on IMDB	Acc on AG-News	dim
`GloVe.6B.300d`	0.824	0.897	300
`e5-base-v2`	0.924	0.916	768
`text-embedding-ada-002`	0.936	0.924	1536
`text-embedding-3-large`	0.964	0.934	3072

Table 2: Downstream performance of different embedding models

5.4 Is task-specific prompt necessary?

We also examinate the necessity of task-specific prompt. As illustrated in Fig 2, the "task-specific" prompt is optional depending on the actual downstream task. We compare the performance of distilled data with and without the prompt, and result is shown in Table 3. We observed that **the optional task-specific prompt brings additional benefit to the distillation**. The effect is more noticeable on larger embedding model (like `text-embedding-3-large`), and almost no effect on smaller models (like `GloVe.6B.300d` and `e5-base-v2`).

Model	Without Prompt	With Prompt
`GloVe.6B.300d`	0.773	0.777
`e5-base-v2`	0.795	0.793
`text-embedding-ada-002`	0.789	0.819
`text-embedding-3-large`	0.790	0.825

Table 3: The averaged accuracy with and without task-specific prompt, evaluated on AG-News with distillation ratio of 0.1%.

5.5 Is vec2text necessary?

The vec2text process plays a crucial role in converting embedding vectors into transferable plain text. This raises an important question: Is such a process truly indispensable? One alternative method involves employing kNN (k-nearest neighbors) rather than k-centroid during the sampling phase. This adjustment leads to the approach converging on a core-set selection strategy, thereby rendering the vec2text step redundant. In our experiments with this alternative configuration, we observed that the average accuracy using "core-set" samples decreased by approximately 5% in comparison to our proposed method.

5.6 Effect of the vec2text model's quality

We explore the relationship between the performance of the vec2text model and the quality of the distilled text it generates. Adhering to the framework outlined in (Morris et al., 2023), we employ accuracy as a metric to evaluate the vec2text model within the same domain. A higher accuracy indicates that the vec2text model is more capable of faithfully reconstructing text from embeddings. To examine this relationship, we trained the vec2text model for varying numbers of epochs to produce several instances of the model, each with differing levels of accuracy, and then assessed the quality of the text they generated. As demonstrated in Figure 3, our findings suggest the existence of an optimal vec2text model configuration that yields the highest quality of generated text. Our analysis reveals that when the accuracy is low, the vec2text

model is likely under-trained, resulting in distilled text that fails to effectively encapsulate the information present in the embedding space. Conversely, when the model achieves high accuracy, it may be over-trained, causing the generated text to regress towards mirroring the original data too closely and thus failing to fulfill the objective of information condensation.

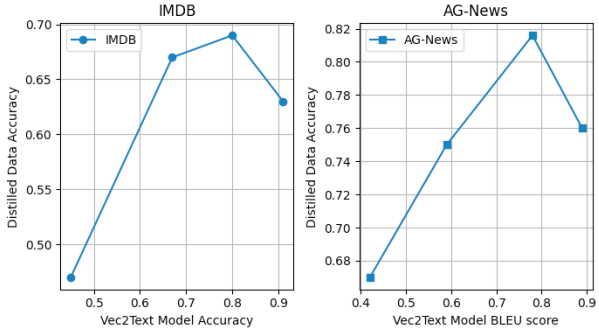

Figure 3: Comparative Analysis of Vec2Text Model Performance on Text Distillation.

5.7 Comparison with LoRA

Our proposed method falls into the category of data efficiency, whereas there are parameter-efficient methods, such as LoRA, that focus on reducing the number of trainable parameters. To compare these distinct approaches, we trained a T5-base model using different methods. The detailed results in Table 4 and a visualization plot is in Figure 1. Compared to LoRA, dataset distillation achieves similar accuracy with much faster training speeds. While LoRA is parameter-efficient, its training time does not decrease proportionally to the reduction in trainable parameters, primarily due to data and computational overhead. This distinction underscores the unique advantage of dataset distillation in reducing training time, which can be crucial in applications requiring extremely fast iterations.

	IMDB		AG News	
	Time	Acc.	Time	Acc.
Full Data	3247	0.936	1391	0.956
LoRA(0.8% params), full data	2862	0.893	11483	0.914
LoRA(0.4% params), full data	2461	0.887	10351	0.908
random data, 1% of full data	66	0.765	506	0.791
distilled data, 1% of full data	67	0.891	508	0.936

Table 4: Comparison between dataset distillation and LoRA. Time is the training time in seconds.

5.8 Cost of Generating Embedding Vectors

For commercial embedding models (text-embedding-ada-002 and text-embedding-3-large), the API cost is cheap. For instance, generating full embeddings for all 120k AG-News data costs about $1.7 based on current pricing.[6].

6 Limitation

If we revisit the text distillation problem, our proposed framework introduces an encoder-decoder architecture to utilize the existing powerful embedding model. Similar to the image compression problem with a encoder-decoder model, the meaningful and valuable information are condensed on a manifold in the high dimensional embedding space. The original data are discretely distributed on the manifold, and the sampling process is picking "synthetic" embedding vectors that contains more information. Then, in order to generate transferable textual data, the vec2text model reconstructs plain-text data with these embedding vectors. From an informational perspective, the sampling stage represents a process of **compressing** information, whereas the vec2text phase involves the **loss** of information to facilitate transferability.

While our proposed method demonstrates significant potential in dataset distillation for NLP tasks, it is not without its challenges. Extending our approach to tasks beyond classification, such as Semantic Text Similarity (STS), requires a nuanced redesign of the sampling and vec2text inversion stages due to the complex relationships inherent in STS data formats. Moreover, crafting a versatile vec2text model that is both high-quality and broadly applicable across different tasks remains a demanding endeavor, constrained by the specificity of task and dataset requirements. Despite these limitations, our work opens up promising avenues for future research, notably in exploring the use of Encoder-Decoder models like T5 as universal vec2text solutions, and in integrating our framework with bi-level optimization methods for enhanced dataset distillation. These directions not only hold the promise of overcoming current constraints but also advancing the efficiency and applicability of dataset distillation in NLP.

[6]https://openai.com/blog/new-embedding-models-and-api-updates

7 Conclusion

We introduced a novel dataset distillation framework for textual data that utilizes language model embeddings to create compact, highly informative datasets. Our experiments demonstrate that the distilled text maintains high accuracy in classification tasks and is compatible with various downstream architectures. Ablation studies highlight critical factors affecting distillation quality, such as the embedding model quality, task-specific prompts, and vec2text model training.

References

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. 2019. Gradient based sample selection for online continual learning. *Advances in neural information processing systems*, 32.

Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. 2017. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495.

Ondrej Bohdal, Yongxin Yang, and Timothy Hospedales. 2020. Flexible dataset distillation: Learn labels instead of images.

Stephen R Bond, Anke Hoeffler, and Jonathan RW Temple. 2001. Gmm estimation of empirical growth models. *Available at SSRN 290522*.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

George Cazenavette, Tongzhou Wang, Antonio Torralba, Alexei A. Efros, and Jun-Yan Zhu. 2022. Dataset distillation by matching training trajectories.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Geoffrey Cideron, Sertan Girgin, Anton Raichuk, Olivier Pietquin, Olivier Bachem, and Léonard Hussenot. 2022. vec2text with round-trip translations. *arXiv preprint arXiv:2209.06792*.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. 2019. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*.

Jiawei Du, Yidi Jiang, Vincent Y. F. Tan, Joey Tianyi Zhou, and Haizhou Li. 2023. Minimizing the accumulated trajectory error to improve dataset distillation.

Dan Feldman. 2020. Introduction to core-sets: an updated survey. *arXiv preprint arXiv:2011.09384*.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

Chengcheng Guo, Bo Zhao, and Yanbing Bai. 2022. Deepcore: A comprehensive library for coreset selection in deep learning. In *International Conference on Database and Expert Systems Applications*, pages 181–195. Springer.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Krishnateja Killamsetty, Sivasubramanian Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. 2021a. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, pages 5464–5474. PMLR.

Krishnateja Killamsetty, Xujiang Zhao, Feng Chen, and Rishabh Iyer. 2021b. Retrieve: Coreset selection for efficient and robust semi-supervised learning. *Advances in Neural Information Processing Systems*, 34:14488–14501.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Saehyung Lee, Sanghyuk Chun, Sangwon Jung, Sangdoo Yun, and Sungroh Yoon. 2022. Dataset condensation with contrastive signals.

Shiye Lei and Dacheng Tao. 2023. A comprehensive survey of dataset distillation.

Guang Li, Bo Zhao, and Tongzhou Wang. 2022. Awesome dataset distillation. `https://github.com/Guang000/Awesome-Dataset-Distillation`.

Yongqi Li and Wenjie Li. 2021. Data distillation for text classification. *arXiv preprint arXiv:2104.08448*.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Vahab Mirrokni and Morteza Zadimoghaddam. 2015. Randomized composable core-sets for distributed submodular maximization. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 153–162.

John X Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M Rush. 2023. Text embeddings reveal (almost) as much as text. *arXiv preprint arXiv:2310.06816*.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Timothy Nguyen, Zhourong Chen, and Jaehoon Lee. 2021. Dataset meta-learning from kernel ridge-regression.

Timothy Nguyen, Roman Novak, Lechao Xiao, and Jaehoon Lee. 2022. Dataset distillation with infinitely wide convolutional networks.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Ozan Sener and Silvio Savarese. 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*.

Tianxiao Shen, Jonas Mueller, Regina Barzilay, and Tommi Jaakkola. 2020. Educating text autoencoders: Latent representation guidance via denoising. In *International conference on machine learning*, pages 8719–8729. PMLR.

Ilia Sucholutsky and Matthias Schonlau. 2021. Soft-label dataset distillation and text dataset distillation. In *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE.

Kai Wang, Bo Zhao, Xiangyu Peng, Zheng Zhu, Shuo Yang, Shuo Wang, Guan Huang, Hakan Bilen, Xinchao Wang, and Yang You. 2022. Cafe: Learning to condense dataset by aligning features.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A. Efros. 2020. Dataset distillation.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, and Ming Zhou. 2020. MIND: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3597–3606, Online. Association for Computational Linguistics.

Wenpeng Yin, Dragomir Radev, and Caiming Xiong. 2021. Docnli: A large-scale dataset for document-level natural language inference.

Bo Zhao and Hakan Bilen. 2022. Dataset condensation with distribution matching.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. 2021. Dataset condensation with gradient matching.

A Architecture of Downstream Models

A.1 LR, NB and SVM

We utilize the implementation from **scikit-learn** with the following imports:

```
\begin{verbatim}
from sklearn.linear_model import
    ↪ LogisticRegression
from sklearn.naive_bayes import
    ↪ MultinomialNB
from sklearn.svm import LinearSVC
\end{verbatim}
```

A.2 TextCNN

We follow the definition in (Sucholutsky and Schonlau, 2021) to have a fair comparison between TDD and our method. Detail is below:

```
%\begin{verbatim}
class TextCNN(nn.Module):
    def __init__(self, vocab_size,
        ↪ embedding_dim, num_filters,
        ↪ filter_sizes, output_dim,
        ↪ dropout,
        ↪ pretrained_embeddings):
        super(TextCNN, self).__init__()
        self.embedding = nn.Embedding(
            ↪ vocab_size, embedding_dim
            ↪ )
        self.embedding.weight.data.copy_
            ↪ (torch.from_numpy(
            ↪ pretrained_embeddings))
        self.embedding.weight.
            ↪ requires_grad = False
        self.convs = nn.ModuleList([
            nn.Conv2d(in_channels=1,
                ↪ out_channels=
                ↪ num_filters,
                ↪ kernel_size=(fs,
                ↪ embedding_dim))
            for fs in filter_sizes
        ])
        self.fc = nn.Linear(num_filters
            ↪ * len(filter_sizes),
            ↪ output_dim)
        self.dropout = nn.Dropout(
            ↪ dropout)

    def forward(self, x):
        #x = x.permute(1, 0)
        embedded = self.embedding(x)
        embedded = embedded.unsqueeze(1)
        conved = [F.relu(conv(embedded))
            ↪ .squeeze(3) for conv in
            ↪ self.convs]
        pooled = [F.max_pool1d(conv,
            ↪ conv.shape[2]).squeeze(2)
            ↪  for conv in conved]
        cat = self.dropout(torch.cat(
            ↪ pooled, dim=1))
        return self.fc(cat)
%\end{verbatim}
```

A.3 TextRNN

Similar to TextCNN, we change the strucutre to RNN like below:

```
class TextRNN(nn.Module):
    def __init__(self, vocab_size,
        ↪ embedding_dim, hidden_dim,
        ↪ output_dim, n_layers,
                bidirectional, dropout,
                    ↪
                    ↪ pretrained_embeddings
                    ↪ ):
        super(TextRNN, self).__init__()
        # Embedding layer
        self.embedding = nn.Embedding(
            ↪ vocab_size, embedding_dim
            ↪ )
        self.embedding.weight.data.copy_
            ↪ (torch.from_numpy(
            ↪ pretrained_embeddings))
        self.embedding.weight.
            ↪ requires_grad = False
        # RNN layer
        self.rnn = nn.LSTM(embedding_dim
            ↪ , hidden_dim, num_layers=
            ↪ n_layers, bidirectional=
            ↪ bidirectional, dropout=
            ↪ dropout, batch_first=True
            ↪ )
        self.fc = nn.Linear(hidden_dim *
            ↪  2 if bidirectional else
            ↪ hidden_dim, output_dim)
        self.dropout = nn.Dropout(
            ↪ dropout)

    def forward(self, text):
        embedded = self.embedding(text)
        output, (hidden, cell) = self.
            ↪ rnn(embedded)
        if self.rnn.bidirectional:
            hidden = self.dropout(torch.
                ↪ cat((hidden[-2,:,:],
                ↪ hidden[-1,:,:]), dim
                ↪ =1))
        else:
            hidden = self.dropout(hidden
                ↪ [-1,:,:])

        return self.fc(hidden)
```

A.4 distilBERT

```
from transformers import
    ↪ DistilBertTokenizer,
    ↪ DistilBertForSequenceClassification
    ↪
model_name = "distilbert-base-uncased"
tokenizer = DistilBertTokenizer.
    ↪ from_pretrained(model_name)
model_config = DistilBertConfig.
    ↪ from_pretrained(model_name,
    ↪ num_labels=2)
model =
    ↪ DistilBertForSequenceClassification
    ↪ .from_pretrained(model_name,
    ↪ config=model_config)
```

A.5 T5-Base

```
from transformers import T5Tokenizer,
    ↪ T5ForConditionalGeneration
model_name = "t5-base"
tokenizer = T5Tokenizer.from_pretrained(
    ↪ model_name)
model = T5ForConditionalGeneration.
    ↪ from_pretrained(model_name)
```

B Implementation Detail

B.1 Training Hyper-parameters

All the training and evaluation are completed within the PyTorch(Paszke et al., 2019) framework on an AWS p3.2xlarge instance, with 16GB GPU memory. Model detail is mentioned in Appendix A. We use AdamW(Loshchilov and Hutter, 2018) as the optimizer and linear learning rate scheduler. The maximum input token length is set to 128 for all datasets. All models are trained for 20 epochs by default to fully converge. The learning rate was chosen within 1e-5, 5e-4, 1e-4, 5e-3 after grid search. Generally, smaller training size requires larger learning rate to converge to the optimal performance.

B.2 K-centroid Sampling

We use the standard KMeans method from the sklearn.cluster library with Euclidean distance.

```
from sklearn.cluster import KMeans

# Perform k-means clustering
kmeans = KMeans(n_clusters=k,
    ↪ random_state=42).fit(
    ↪ embeddings)
# Get the cluster centroids
centroids = kmeans.cluster_centers_
return centroids
```

C Performance of DaLLME under different distillation ratio

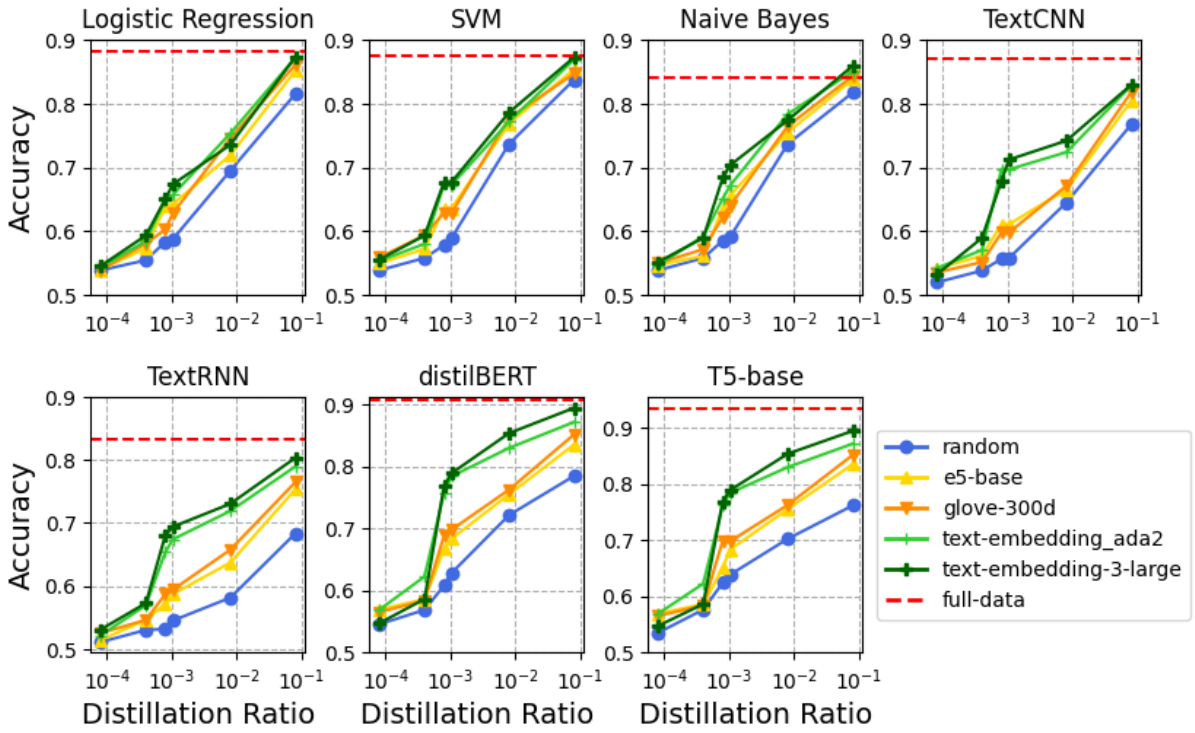

Figure 4: Accuracy vs Distillation Ratio on IMDB.

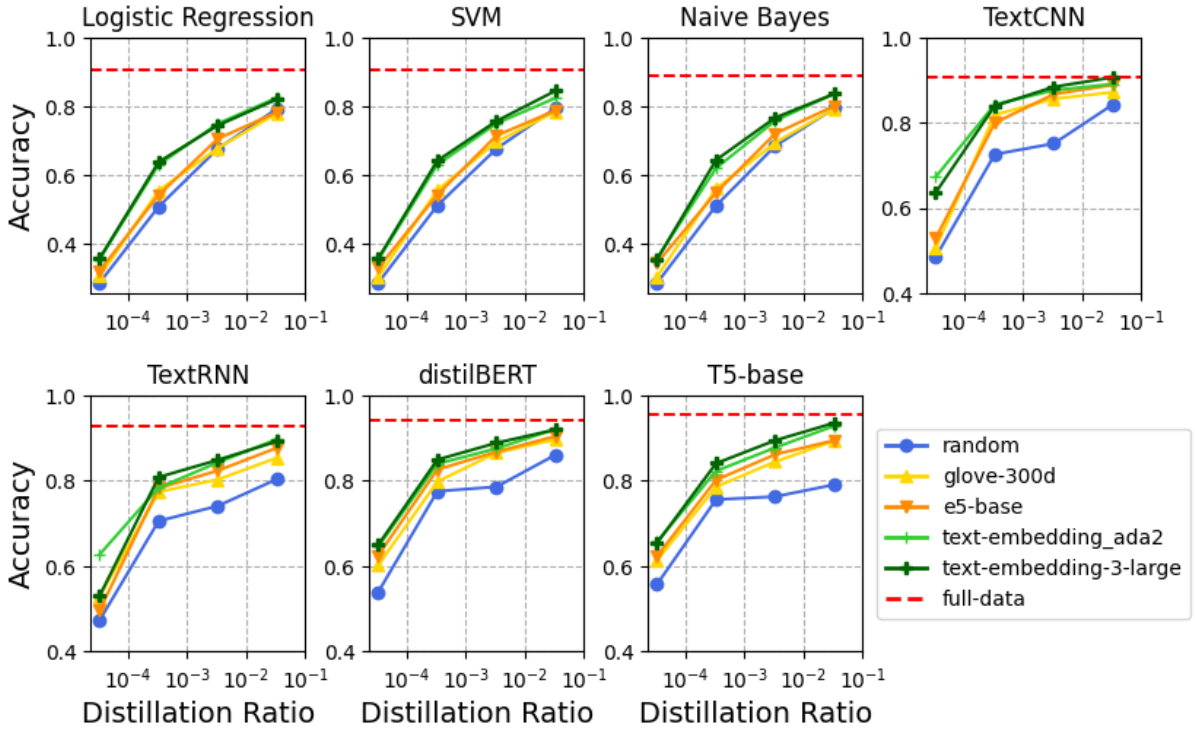

Figure 5: Accuracy vs Distillation Ratio on AG-News.